A deep learning-based strategy for producing dense 3D segmentations from sparsely annotated 2D images

Vijay Venu Thiyagarajan[1], Arlo Sheridan[2], Kristen M. Harris[1,*], Uri Manor[2,3,4,*]

[1]Department of Neuroscience, Center for Learning and Memory, University of Texas at Austin, Austin Texas, 78712
[2]Waitt Advanced Biophotonics Center, Salk Institute for Biological Studies, La Jolla, CA, 92037
[3]Department of Cell & Developmental Biology, School of Biological Sciences, University of California, San Diego, CA 92093
[4]Halıcıoğlu Data Science Institute, University of California, San Diego, CA 92093
[*]Co-corresponding authors: kharris@utexas.edu, uri@ucsd.edu

ABSTRACT
Producing dense 3D reconstructions from biological imaging data is a challenging instance segmentation task that requires significant ground-truth training data for effective and accurate deep learning-based models. Generating training data requires intense human effort to annotate each instance of an object across serial section images. Our focus is on the especially complicated brain neuropil, comprising an extensive interdigitation of dendritic, axonal, and glial processes visualized through serial section electron microscopy. We developed a novel deep learning-based method to generate dense 3D segmentations rapidly from sparse 2D annotations of a few objects on single sections. Models trained on the rapidly generated segmentations achieved similar accuracy as those trained on expert dense ground-truth annotations. Human time to generate annotations was reduced by three orders of magnitude and could be produced by non-expert annotators. This capability will democratize generation of training data for large image volumes needed to achieve brain circuits and measures of circuit strengths.

In 3D instance segmentation, voxels of a 3D image volume are partitioned into distinct object instances, where each voxel is associated with a corresponding object. Instance segmentations of brain neuropil comprising intricate processes of dendrites, axons, and glia, are required for downstream connectomics and ultrastructural studies[1,2]. The voxel assignment in neuropil segmentation is often challenging due to the complex morphologies of the fine processes, which often branch and converge at multiple locations in the volume. Individual flaws in a segmentation can have large consequences in the resultant topology of the segmented processes. For example, if an axonal process was erroneously connected at a branch point to another axon, the resulting connection could project to the wrong target neurons.

Despite the challenges, automatic deep learning-based methods for electron microscopy (EM) segmentation have shown great promise[1–7]. Flood-Filling Networks (FFN) are currently considered to be state-of-the-art[6]; however, the computational resources required to use and train FFNs are unattainable for most laboratories. A different approach involves convolutional neural networks to generate boundary predictions on image volumes, followed by prediction and post-processing to output a 3D instance segmentation[3–5,7,8]. This boundary-based approach requires 100x less computational cost and is easily parallelized, enabling much faster processing of large volumes, but it is usually less accurate than FFNs[5]. Recent work showed that boundary detection-based methods can match the accuracy of FFNs with 10-100x higher efficiency by adding local shape descriptors (LSDs) for auxiliary training[7]. Additionally, there are other efficient methods to generate large segmentations without LSDs, such as boundary predictions with improved post-processing

using mutex watershed or multicut[9,10]. Thus, the implementation of efficient methods provides an important step towards acceleration of 3D segmentation.

The performance of deep learning models is fundamentally dependent on the quality and quantity of the available training data. For segmentation of brain neuropil, ground-truth data typically must be dense and diverse to be useful. Explicitly, all objects in a volume typically must be completely annotated (dense) and sampled from multiple (diverse), representative regions of the target data. Previous studies regarding the accuracy of different deep learning methods used large amounts of ground-truth data for training and evaluation. The annotations in these training data required enormous human effort. For example, the zebra finch EM dataset had 33 densely annotated training volumes, for a total of ~200μm^3 of labeled objects, and 50 manually proofread skeletons totalling 97 mm^6[6,7]. Obtaining this ground-truth dataset required multiple time-consuming refinement rounds by expert annotators (Joergen Kornfeld, personal communication). In another example, the complete morphological reconstruction of 15 Kenyon cells in the adult fly brain, each with a mean cable length of 0.78 mm, took more than 150 human hours[11]. The annotated ground-truth for every dendritic, axonal, and glial process in a rat hippocampal volume of 180μm^3 took more than 1,000 hours of manual annotation and curation[12]. These examples illustrate that the human effort spent generating ground-truth data currently stands as a major bottleneck in the production of useful segmentations and downstream analyses.

We present a new approach that markedly reduces the human effort to generate ground-truth data that can be used to improve machine learning algorithms for automated 3D segmentation. We evaluated the 3D segmentations that were generated as a function of the amount of annotations and found similar accuracy for all amounts, including as little as ten minutes of non-expert annotations on a single image. We also tested whether the generated segmentations can be used without any proofreading as pseudo ground-truth training data for bootstrapping 3D models. We show the method worked for multiple brain neuropil datasets as well as a fluorescence microscopy dataset, thus illustrating its generalizability. We provide a workflow for new users that helps to replace the enormous human effort currently needed to trace and curate experimental 3D electron microscopy datasets. We anticipate that the new tools and techniques will enable most laboratories to generate training data for complex 3D instance segmentation tasks rapidly, even if they lack the otherwise prohibitive compute resources or human expertise.

Results

Deep learning networks can learn to generate dense 3D volumetric predictions from sparsely annotated 2D slices[13]. However, this approach has not yet been extended to highly anisotropic data or to complex 3D instance segmentation that are typical of serial section EM datasets. To tackle this, we have developed an approach in which a human first produces sparse 2d annotation on an image (or subset of images) (Fig. 1). Next, a 2D network is trained on the sparse 2D annotations to make dense predictions on each image. Then these 2D predictions are stacked spatially as input for a separate 3D network, trained using random synthetic 3D data generated on the fly, to infer 3D boundaries from the 2D stacks. Finally, conventional post-processing is done to obtain a 3D segmentation (Fig. 1).

Using this new approach, we present experimental results of bootstrapping neuropil segmentation models with pseudo ground-truth training data generated rapidly by the 2D->3D method. The segmentation models were trained on various amounts and types of sparse annotation. We compared the quality and accuracy of

segmentations generated by 3D LSD models bootstrapped from different amounts and types of annotation. We also compared the total time to generate the segmentations for each level of sparse manual annotation.

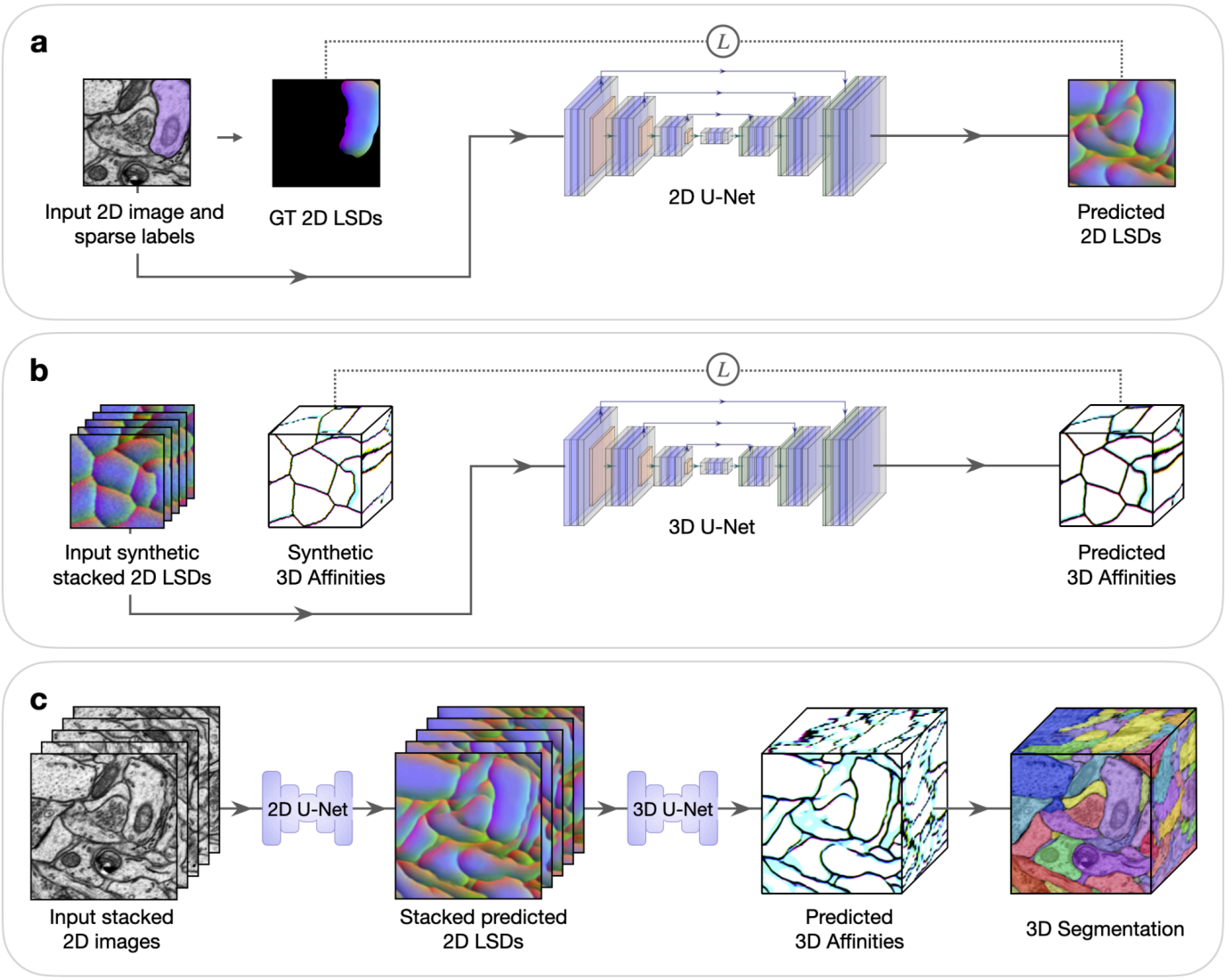


Fig. 1: Method to generate dense 3D instance segmentations from sparse 2D training data. Five example sections are shown in stacked 2D images or LSDs to simplify visualization. LSDs show the first three components as RGB-channels. All networks use the weighted mean-squared-error (MSE) loss function for training, denoted by L. **a,** Sparse 2D to dense 2D. Input 2D images with sparse ground-truth labels are used to train a 2D U-Net to learn dense 2D LSDs. Background regions of the ground-truth 2D LSDs are not used in the computation of the loss during training. **b,** Stacked dense 2D to dense 3D. A 3D U-Net is trained to learn 3D affinities from stacked 2D LSDs using synthetic labels. **c,** Combined 2D to 3D inference pipeline. Sections from a 3D image volume are used as input to the trained 2D U-Net to generate stacked 2D LSDs. The trained 3D U-Net infers 3D affinities from the stacked 2D LSDs. A 3D segmentation is generated from the 3D affinities using standard post-processing techniques.

Datasets

We chose six datasets for these experiments, including HARRIS-15[12], FIB-25[14], CREMI-A,B,C[15], and EPI[16] . Each dataset (except EPI) consisted of two EM volumes with dense annotations, Volume 1 and Volume 2, with dataset names, content descriptions, sizes, number of instances, imaging modalities, and resolutions detailed in Supplementary Table 1. At least two volumes per dataset were required to avoid propagating volume-specific biases. To show this approach extends beyond EM, we included confocal laser scanning microscopy volumes of plant ovules (EPI).

For all datasets, we perform instance-level ablations on the dense labels of Volume 1 to generate different amounts of sparse training data (Supplementary Fig. 1). For every ablation, we chose three mutually exclusive selections of the required number of instances. In addition, we spent 30 minutes to generate non-expert sparse 2D annotations in a single section and another 30 minutes for sparse 3D annotations across a few sections on Volume 1 for each dataset (Supplementary Fig. 2). The 2D and 3D annotations were partitioned into three artificial 10 minute annotations such that labels were mutually exclusive. In addition, we provide results from a subset of ablations (1 object, 2 adjacent objects, 2, 5, 10, 50, or 100 random objects) and compared to the dense ground-truth training. The different ablations and manual annotations tested in the experiments are detailed in Supplementary Note A.

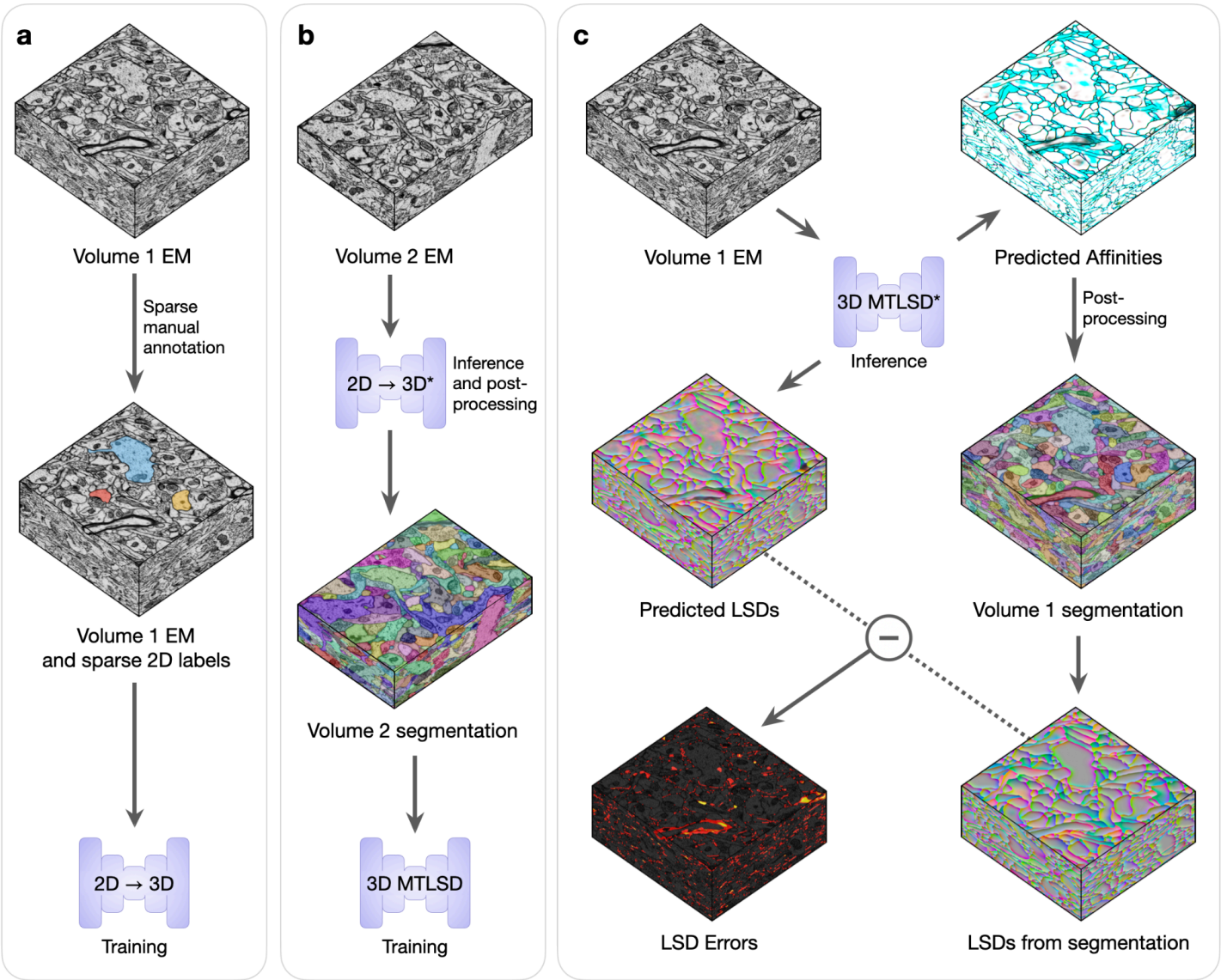


Fig. 2: Bootstrapping dense 3D segmentations. Volumes 1 and 2 are two EM volumes without any annotations. **a,** Sparse 2D ground-truth labels are manually created on Volume 1, which are used to train the 2D to 3D networks. **b,** A 3D segmentation of Volume 2 is generated using the trained 2D to 3D networks (denoted by *) and standard post-processing. The resulting 3D segmentation is used as pseudo ground-truth training data, without any masking or proofreading, for an untrained 3D MTLSD network. **c,** The trained 3D MTLSD network infers 3D affinities and LSDs on Volume 1. Post-processing is applied to the 3D affinities to generate a 3D segmentation on Volume 1, from which LSDs are computed. The element-wise difference between the model's LSDs and the computed segmentation LSDs generates a heat map of errors. These errors can be thresholded to obtain a mask of high-error regions, which can be used for filtering of segmentations and targeted refining during subsequent training.

For every combination of dataset and initial training data, we conducted the experimental procedure as described in Fig. 2 and Supplementary Note B. Briefly, Volume 1's sparse annotations were used to train the

2D->3D method and then segment Volume 2. The generated segmentation of Volume 2 was used to train a 3D model, which was then used to segment Volume 1. Inference and post-processing steps were conducted in a grid-fashion to explore a range of values for parameters that might alter the output segmentations. We designated the best performing segmentations in the grid as the representative bootstrapping results for that dataset and initial amount of training data.

To evaluate the accuracy of bootstrapped (Volume 1) segmentations as a function of the amount of initial training data, we report segmentation accuracy with the min-cut metric (MCM). The MCM is a skeleton-based measure of the total number of split and merge edit operations that would need to be proofread to ensure accuracy[7]. We adapt it to be the total number of edits needed divided by the total number of objects in the ground-truth. Ground-truth labels were filtered to remove labels smaller than 500 pixels and connected components were relabeled. Ground-truth skeletons were generated as described in Supplementary Note C. We report additional metrics such as variation of information in Supplementary Figures 3-5.

Bootstrapped Segmentations

For all datasets, we found the quality of pseudo ground-truth segmentations and bootstrapped segmentations to be good for all sparsities, including ten minutes of sparse 2D annotation (Fig. 3, Fig. 4, Supp. Fig. 3-5). Importantly, sparse non-expert annotations of a single section yielded bootstrapped segmentations similar in accuracy to dense expert annotations of the volume (Fig. 4). One possible explanation is that ten minutes of annotations in a single image contained enough boundary information between instances that the 2D network learned accurate 2D predictions with significantly less human annotation time.

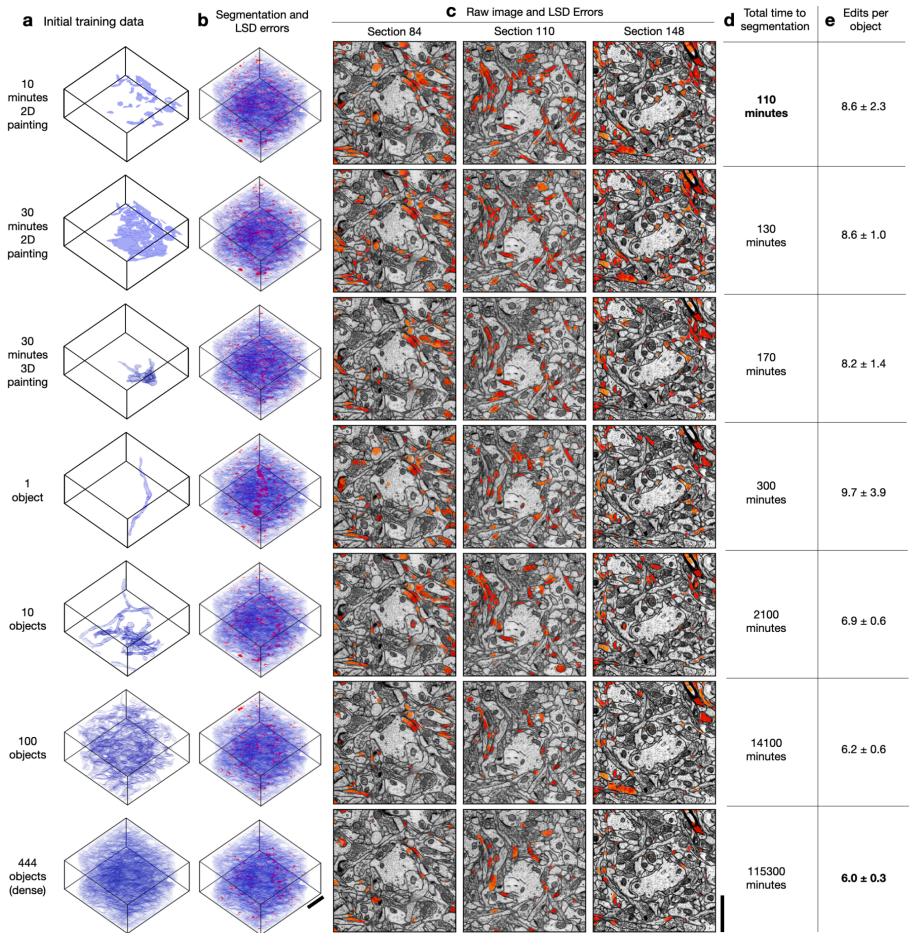


Fig. 3: Example from bootstrapped models for HARRIS-15. a, The first column shows different amounts of initial ground-truth training data used to generate a dense 3D segmentation of the test volume. **b,** Second column shows bootstrapped segmentations in agreement with ground-truth manual annotations (blue) and the LSD split/merge errors (red). The errors visualized are computed as the element-wise difference between the LSDs computed from ground-truth manual annotations and the LSDs computed from the bootstrapped segmentations. **c,** The third column shows a selection of 2D sections of the LSD errors overlaid on raw EM images with red areas showing examples of split or merge errors on that section. **d,** The fourth column shows the total time required to obtain a 3D segmentation starting from no annotations through bootstrapping and is a sum of the times for manual annotation, training 2D->3D, training 3D MTLSD, and inference and post-processing. **e,** The fifth column provides the average number (± standard deviation) of split and merge edits required for each object to match the dense ground-truth skeletons. Scale bars are 1.5 μm.

We note that the quality of bootstrapped segmentations is dependent on the image data and the quality of the generated pseudo ground-truth used for training. The occurrence of high LSD error values along the boundaries of the segmented processes indicate minor pixel-wise differences with the boundaries of the manual ground-truth labels and not a significant difference in the morphology of the process. In Fig. 3, we apply binary opening to the LSD error mask and use it to visualize the LSD errors without the minor boundary differences. This operation is especially useful in the cases where manual annotations are not pixel-precise along boundaries.

We measured the total time to generate a segmentation for all sparsities by summing the estimated human annotation time with the time taken by the machine for all training, inference, and post-processing (Fig. 3). The machine time was ~100 minutes for all datasets and sparsities. A model bootstrapped with 10 minutes of sparse annotations on a single section was able to generate a dense 3D segmentation in about 110 minutes. In contrast, a model bootstrapped from dense annotations made with 1000x more human annotation time generated a segmentation requiring similar proofreading effort, with at most 2-3 fewer edits per path length in microns (Fig. 4). We conclude that densely annotated ground-truth is not necessary to provide a starting point for training dedicated 3D models. Instead, sparse annotations are sufficient for bootstrapping automatic neuropil segmentation and require about the same amount of proofreading for marked reduction in total time.

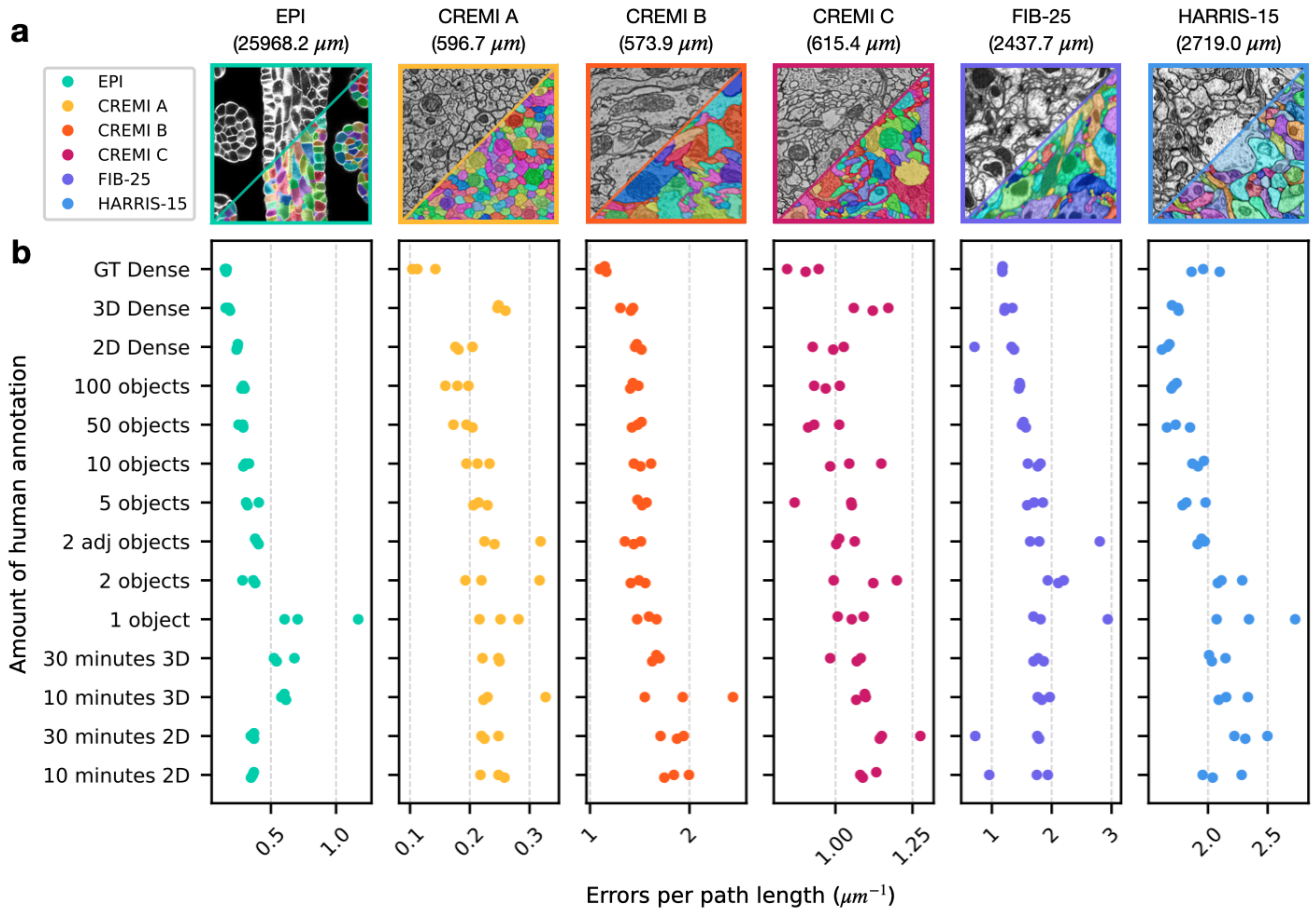


Fig. 4: Quantitative results of 3D segmentations bootstrapped from sparse annotations. a, Example images and ground-truth labels from each dataset and color-coding key. The total path length of the dense ground-truth skeletons in each training volume is indicated in parenthesis. **b,** Number of split and merge errors needing correction per skeleton path length to match the dense ground-truth skeletons. Lower scores are better. Each dataset had three separate tests for each amount of annotation (illustrated as three dots of the same color per row). Dense segmentations of a separate test volume (pseudo ground-truth) were first created by 2D->3D models trained on different amounts of initial ground truth annotations, which are sorted from bottom to top of the y-axis in order of increasing human annotation effort. Scores were then computed by comparing ground-truth annotations to segmentations produced by a 3D model trained on the pseudo ground-truth segmentation. 2D (or 3D) Dense refers to training the initial 2D->3D (or 3D) model on all available 2D (or 3D) annotations to generate pseudo ground-truth. GT Dense refers to directly training a 3D model on ground-truth annotations of the test volume to generate the segmentation of the initial training volume.

The new algorithms are available at github (github.com/ucsdmanorlab/bootstrapper) to generate dense 3D segmentations from sparse 2D annotations and bootstrap 3D models. We implemented a Napari plugin to allow users to apply the 2D->3D method to generate and export dense volumetric segmentations from minimal annotations made through Napari's graphical user interface (github.com/ucsdmanorlab/napari-bootstrapper). Our plugin, along with pre-trained models and plugins like Segment Anything Model (SAM) fine-tuned on microscopy data, can reduce manual 2D annotation to a few clicks[17,18].

Discussion

Machine learning models depend on sufficient quantities of high quality training data. Generating training data for segmentation of multidimensional datasets, e.g., 3D neuropil segmentation, is prohibitively time consuming to generate and proofread manually. This is largely due to the difficulty in segmenting 3D structures on digital display screens that can only display one 2D image at a time. These physical limitations coupled with the need for densely annotated data from a diverse range of example images that represent the full landscape of highly variable features renders 3D annotation a monumental task that most researchers and laboratories are not capable of within a reasonable timeframe. The total resulting cost of generating manually annotated data in both human hours and research dollars ultimately presents a barrier towards incalculable new discoveries. Thus, there is an urgent need to develop new tools and methods that can massively accelerate and democratize the generation of ground-truth data and automated segmentation.

Here we have demonstrated the novel use of a 2D->3D method to generate dense volume segmentations rapidly from sparse 2D annotations for complex instance segmentation tasks. We also demonstrated that rapidly generated segmentations provide sufficiently accurate pseudo ground-truth training data for bootstrapping 3D segmentation models. These lightweight models do not perform as well as a dedicated 3D model trained on dense and diverse ground-truth annotations. However, the 100-1000x shorter amount of time required to segment many complex subvolumes far outweighs the cost of training human annotators to produce accurate dense manual annotations for use in dedicated 3D models. Furthermore, the relative ease of refining a dedicated model trained on such subvolumes will ultimately result in automation of better large scale instance segmentations. In short, this approach enables massively accelerated convergence on ground-truth annotations at a fraction of the cost, and is effective for even the most complex instance segmentation tasks such as neuropil segmentation.

Generating dense volumetric instance segmentations from sparse image annotations has been demonstrated for simpler biomedical image segmentation tasks by generalist algorithms[13,19–22], but not for complex instance segmentation tasks like neuropil segmentation. Our method works reliably on small volumes for all initial amounts of sparse image annotations and all investigated datasets, suggesting general applicability. The use of pseudo, incomplete, or imprecise labels as training data is also not novel; neither is the idea of iterative bootstrapping to refine a model by acquiring more segmented data for training[23–26]. Others recently demonstrated an end-to-end pipeline that learned to correct swift and incomplete annotations like skeletons and seeds[27]. Previous work has also addressed the dense annotation bottleneck with the use of pixel embeddings to produce sparse instance masks generated on the fly during training to achieve weak positive-unlabeled supervision[28]. Our use of LSDs enhances these approaches by quickly providing high error regions that streamline proofreading and facilitate automatic refinement of annotations. Thus, LSD-based models make quickly generating ground-truth a practical possibility for large 3D datasets.

In addition to the increased computational efficiency of LSD-based networks, the mask of high error regions from the LSD errors has many valuable applications. A natural application is its use in a targeted proofreading tool. Future work could guide the tool or user to locate potential errors[29–33]. Automatic error correction methods could also learn to correct errors from LSD errors and prompt the user for corrections that require a single click[27,34,35]. Additionally, the mask of high errors can be used for targeted refining of a model during bootstrapping; masked regions in the field of view can be ignored in the loss computation while forcing the model to encounter more FOVs from high error regions.

There are natural future directions for this work. An autocontext approach could be adopted for the 2D->3D method, where the raw EM volume along with the stacked 2D predictions would be inputs to the 3D network[36]. We hypothesize this modification would resolve some ambiguities when extrapolating connectivity between predictions of adjacent sections of the image volume. However, this modification would require dataset-specific training for the 3D network, more diverse and neuron-inspired synthetic labels for training the 3D network, and the use of long-range affinities and mutex watershed for segmentation[4,9].

Together, the presented LSD-based bootstrapping method greatly democratizes the generation and use of annotation data for automated segmentation. This is but a step in the right direction—future work should continue to prioritize approaches that require minimal manual annotation, i.e., self-supervised and unsupervised learning approaches can be integrated into bootstrapping workflows to accelerate even further progress towards automated image segmentation and analysis, thereby enabling scientists to focus more on new discoveries and insights.

Methods

All network variations, training pipelines, and parameter values explored in the post-processing grid-searches are described in detail in the Supplementary Notes D, E, and Supplementary Tables 1-6. GPU nodes with NVIDIA A100s were used for training and inference. Texas Advanced Computing Center (TACC)'s Launcher software was used to run the post-processing and evaluation jobs on TACC's Lonestar6[37]. Hyperparameter optimization was done through a grid search as described in Supplementary Note B.

Importantly, a super computer is not needed to generate new 3D segmentations with our lightweight algorithms. We used the trained 2D U-Net to generate dense 2D predictions for every section of a volume. Then, we stacked the predictions and used them as input to a lightweight 3D U-Net we pre-trained using synthetic 3D data to output 3D affinities (Supplementary Note C). In practice, images from different modalities could adapt the same model since it only uses stacked 2D predictions as input and not the stacked raw images. Since the networks are lightweight, they could be trained concurrently with less than 6 GB of vRAM.

Sparse to Dense

The general principle is that 2D annotations are less time consuming to generate than 3D annotations, as are sparse annotations compared to dense annotations. To build a successful 3D instance segmentation model for neuropil, dense ground-truth training data is typically recommended. We developed a lightweight method to generate dense 3D instance segmentations rapidly by using sparse 2D annotations for training. The generated segmentations can be used as pseudo ground-truth to bootstrap and iteratively refine a dedicated 3D model.

Given EM images with sparse annotations, one can train a 2D U-Net[38] to learn dense cell boundaries, embeddings like LSDs, or both. We used direct-neighbor affinity graphs (affinities) to represent the plasma

membrane boundaries by capturing the connectivity (or affinity value) of every pixel to its immediate neighbors[3]. This representation, compared to binary boundary representations, resolves ambiguities when partitioning voxels into different instances in 3D.

During training, random fields of views of the annotated image(s) were chosen, augmentations were applied, and 2D target predictions were computed from the sparse annotations. The areas lacking annotations were masked out during the computation of the loss between the model's output prediction and the target prediction. This masking restricted the supervision during learning to only the labeled areas. Thus, the model could infer predictions in the unlabeled areas using what it learned from the labeled areas.

Bootstrapping neuron segmentation

We used the 3D Multi-task LSD (3D MTLSD) network for bootstrapping a neuropil segmentation model from pseudo ground-truth data. For small volumes, the addition of LSDs as an auxiliary learning task for affinities achieves comparable accuracy to the current state of the art, while being 100x more computationally efficient[7]. The higher speed of these MTLSD networks is invaluable in this context given the potential need for repeated iterations of generating and proofreading segmentations.

Affinities from the trained MTLSD model can be used to segment new volumes or refine existing segmentations. LSDs from the trained MTLSD model can be used to generate an error map by taking the difference between the predicted 3D LSDs and the 3D LSDs computed from the 3D affinities' segmentation. Applying a simple threshold to the error map can produce a binary mask of high error regions, disregarding minor pixel-wise differences and preserving morphological errors. This mask can drive the model to learn from high error regions in subsequent bootstrapping rounds.

Post-processing

We used our published methods[5,7] to generate dense 3D instance segmentations from affinities. First, we thresholded predicted affinities to generate a binary mask, from which we computed a distance transform and identified local maxima. We used the maxima as seeds for a watershed algorithm to generate an oversegmentation (resulting in supervoxels). Each supervoxel center of mass was stored as a node with coordinates in a region adjacency graph. All nodes of touching supervoxels were connected by edges and added to this graph. In a subsequent agglomeration step, edges were merged hierarchically using the underlying predicted affinities as weights, in order of decreasing affinity.

Acknowledgements

U.M. is supported by the Goeddel Family Technology Sandbox at UCSD, NIA P30AG068635 (Nathan Shock Center), Core Grant application NCI CCSG (CA014195), NIDCD R01 DC021075, NSF NeuroNex Award (2014862), and the CZI Imaging Scientist Award DOI https://doi.org/10.37921/694870itnyzk from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation (funder DOI 10.13039/100014989). K.M.H. and V.V.T. are supported by NIH R01MH095980, NSF NeuroNex Technology Hub Award (1707356), NSF NeuroNex Award (2014862), and NSF NCS Award (2219864).

The authors thank all the manual annotators for their valuable work. Special acknowledgement to Patrick H. Parker for expert curation and proofreading of the HARRIS-15 dataset. The authors also acknowledge James Carson and the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing

HPC resources that have contributed to the research results reported within this paper. URL: http://www.tacc.utexas.edu

References

1. Jain, V., Seung, H. S. & Turaga, S. C. Machines that learn to segment images: a crucial technology for connectomics. *Curr. Opin. Neurobiol.* **20**, 653–666 (2010).
2. Aswath, A., Alsahaf, A., Giepmans, B. N. G. & Azzopardi, G. Segmentation in large-scale cellular electron microscopy with deep learning: A literature survey. *Med. Image Anal.* **89**, 102920 (2023).
3. Turaga, S. C. *et al.* Convolutional Networks Can Learn to Generate Affinity Graphs for Image Segmentation. *Neural Comput.* **22**, 511–538 (2010).
4. Lee, K., Zung, J., Li, P., Jain, V. & Seung, H. S. Superhuman Accuracy on the SNEMI3D Connectomics Challenge. Preprint at https://doi.org/10.48550/arXiv.1706.00120 (2017).
5. Funke, J. *et al.* Large Scale Image Segmentation with Structured Loss Based Deep Learning for Connectome Reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* **41**, 1669–1680 (2019).
6. Januszewski, M. *et al.* High-precision automated reconstruction of neurons with flood-filling networks. *Nat. Methods* **15**, 605–610 (2018).
7. Sheridan, A. *et al.* Local shape descriptors for neuron segmentation. *Nat. Methods* **20**, 295–303 (2023).
8. Nguyen, T. M. *et al.* Structured cerebellar connectivity supports resilient pattern separation. *Nature* **613**, 543–549 (2023).
9. Wolf, S. *et al.* The Mutex Watershed: Efficient, Parameter-Free Image Partitioning. in *Computer Vision – ECCV 2018* (eds. Ferrari, V., Hebert, M., Sminchisescu, C. & Weiss, Y.) 571–587 (Springer International Publishing, Cham, 2018). doi:10.1007/978-3-030-01225-0_34.
10. Pape, C. *et al.* Solving Large Multicut Problems for Connectomics via Domain Decomposition. in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* 1–10 (2017). doi:10.1109/ICCVW.2017.7.
11. Zheng, Z. *et al.* A Complete Electron Microscopy Volume of the Brain of Adult Drosophila melanogaster. *Cell* **174**, 730-743.e22 (2018).
12. Harris, K. M. *et al.* A resource from 3D electron microscopy of hippocampal neuropil for user training and tool development. *Sci. Data* **2**, 150046 (2015).

13. Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T. & Ronneberger, O. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016* (eds. Ourselin, S., Joskowicz, L., Sabuncu, M. R., Unal, G. & Wells, W.) 424–432 (Springer International Publishing, Cham, 2016). doi:10.1007/978-3-319-46723-8_49.

14. Takemura, S. *et al.* Synaptic circuits and their variations within different columns in the visual system of Drosophila. *Proc. Natl. Acad. Sci.* **112**, 13711–13716 (2015).

15. CREMI. https://cremi.org/.

16. Wolny, A. *et al.* Accurate and versatile 3D segmentation of plant tissues at cellular resolution. *eLife* **9**, e57613 (2020).

17. Archit, A. *et al.* Segment Anything for Microscopy. 2023.08.21.554208 Preprint at https://doi.org/10.1101/2023.08.21.554208 (2023).

18. Kirillov, A. *et al.* Segment Anything. Preprint at https://doi.org/10.48550/arXiv.2304.02643 (2023).

19. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. Cellpose: a generalist algorithm for cellular segmentation. *Nat. Methods* **18**, 100–106 (2021).

20. Sugawara, K. *Training Deep Learning Models for Cell Image Segmentation with Sparse Annotations*. http://biorxiv.org/lookup/doi/10.1101/2023.06.13.544786 (2023) doi:10.1101/2023.06.13.544786.

21. Conrad, R., Lee, H. & Narayan, K. Enforcing Prediction Consistency Across Orthogonal Planes Significantly Improves Segmentation of FIB-SEM Image Volumes by 2D Neural Networks. *Microsc. Microanal.* **26**, 2128–2130 (2020).

22. Zhou, F. Y. *et al.* A general algorithm for consensus 3D cell segmentation from 2D segmented stacks. 2024.05.03.592249 Preprint at https://doi.org/10.1101/2024.05.03.592249 (2024).

23. Li, J., Udupa, J. K., Tong, Y., Wang, L. & Torigian, D. A. Segmentation Evaluation with Sparse Ground Truth Data: Simulating True Segmentations as Perfect/Imperfect as Those Generated by Humans. *Med. Image Anal.* **69**, 101980 (2021).

24. Takaya, E., Takeichi, Y., Ozaki, M. & Kurihara, S. Sequential semi-supervised segmentation for serial electron microscopy image with small number of labels. *J. Neurosci. Methods* **351**, 109066 (2021).

25. Matskevych, A., Wolny, A., Pape, C. & Kreshuk, A. From Shallow to Deep: Exploiting Feature-Based Classifiers for Domain Adaptation in Semantic Segmentation. *Front. Comput. Sci.* **4**, (2022).

26. Wang, H. *et al.* Using Unreliable Pseudo-Labels for Label-Efficient Semantic Segmentation. Preprint at http://arxiv.org/abs/2306.02314 (2023).

27. Chen, M. *et al.* Learning to Correct Sloppy Annotations in Electron Microscopy Volumes. in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* 4273–4284 (IEEE, Vancouver, BC, Canada, 2023). doi:10.1109/CVPRW59228.2023.00450.

28. Wolny, A., Yu, Q., Pape, C. & Kreshuk, A. Sparse Object-level Supervision for Instance Segmentation with Pixel Embeddings. in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 4392–4401 (IEEE, New Orleans, LA, USA, 2022). doi:10.1109/CVPR52688.2022.00436.

29. Joyce, J. *et al.* A Novel Semi-automated Proofreading and Mesh Error Detection Pipeline for Neuron Extension. *bioRxiv* 2023.10.20.563359 (2023) doi:10.1101/2023.10.20.563359.

30. Plaza, S. M. Focused Proofreading to Reconstruct Neural Connectomes from EM Images at Scale. in *Deep Learning and Data Labeling for Medical Applications* (eds. Carneiro, G. et al.) 249–258 (Springer International Publishing, Cham, 2016). doi:10.1007/978-3-319-46976-8_26.

31. Haehn, D. *et al.* Design and Evaluation of Interactive Proofreading Tools for Connectomics. *IEEE Trans. Vis. Comput. Graph.* **20**, 2466–2475 (2014).

32. Celii, B. *et al.* NEURD: automated proofreading and feature extraction for connectomics. 2023.03.14.532674 Preprint at https://doi.org/10.1101/2023.03.14.532674 (2023).

33. Dorkenwald, S. *et al.* FlyWire: online community for whole-brain connectomics. *Nat. Methods* **19**, 119–128 (2022).

34. Haehn, D., Kaynig, V., Tompkin, J., Lichtman, J. & Pfister, H. Guided Proofreading of Automatic Segmentations for Connectomics. in 9319–9328 (2018). doi:10.1109/CVPR.2018.00971.

35. Zhao, T., Olbris, D. J., Yu, Y. & Plaza, S. M. NeuTu: Software for Collaborative, Large-Scale, Segmentation-Based Connectome Reconstruction. *Front. Neural Circuits* **12**, 101 (2018).

36. Tu, Z. & Bai, X. Auto-Context and Its Application to High-Level Vision Tasks and 3D Brain Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**, 1744–1757 (2010).

37. Wilson, L. A. & Fonner, J. M. Launcher: A Shell-based Framework for Rapid Development of Parallel Parametric Studies. in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment* 1–8 (Association for Computing Machinery, New York, NY, USA, 2014).

doi:10.1145/2616498.2616534.

38. Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (eds. Navab, N., Hornegger, J., Wells, W. M. & Frangi, A. F.) 234–241 (Springer International Publishing, Cham, 2015). doi:10.1007/978-3-319-24574-4_28.

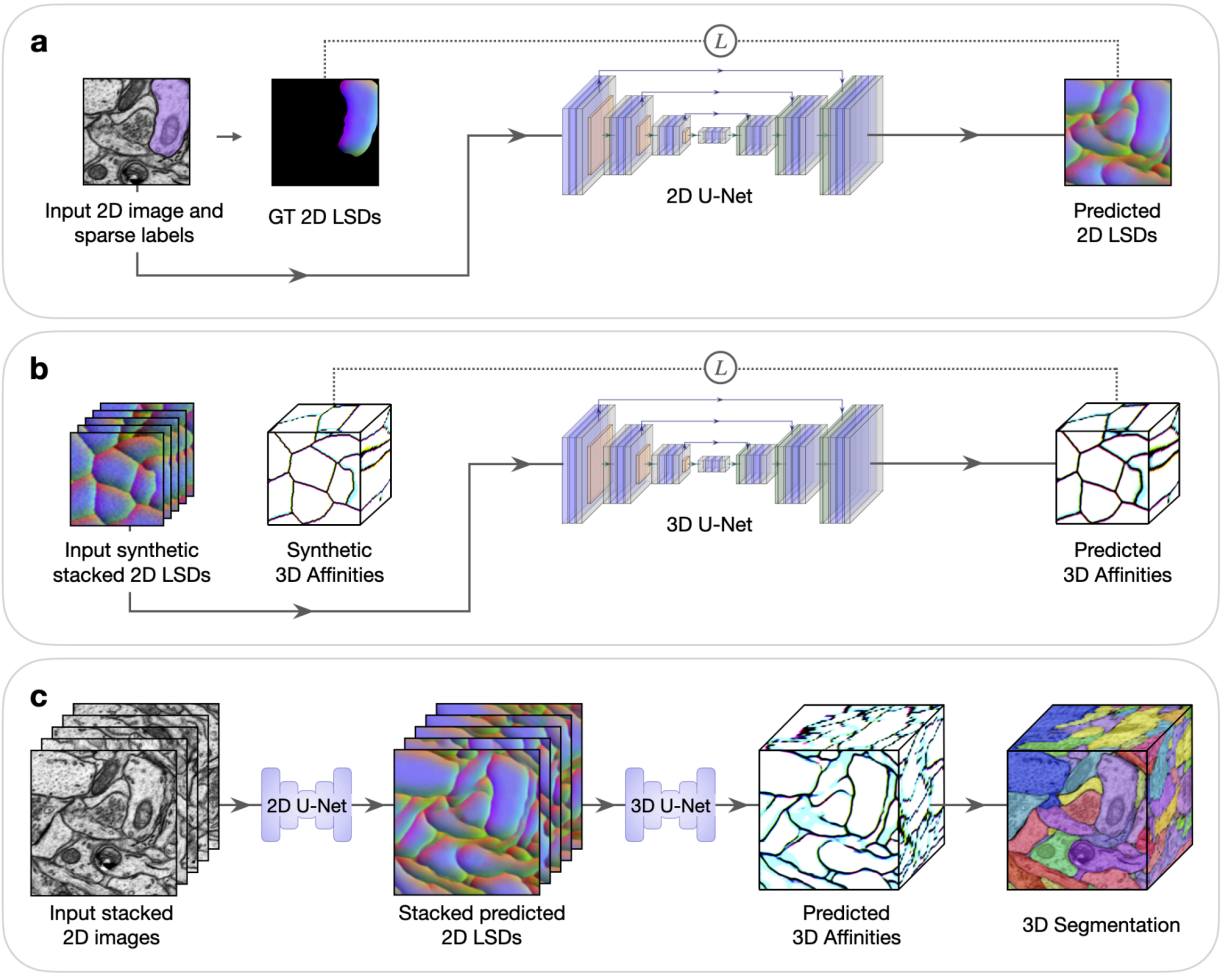


Fig. 1: Method to generate dense 3D instance segmentations from sparse 2D training data. Five example sections are shown in stacked 2D images or LSDs to simplify visualization. LSDs show the first three components as RGB-channels. All networks use the weighted mean-squared-error (MSE) loss function for training, denoted by L. **a,** Sparse 2D to dense 2D. Input 2D images with sparse ground-truth labels are used to train a 2D U-Net to learn dense 2D LSDs. Background regions of the ground-truth 2D LSDs are not used in the computation of the loss during training. **b,** Stacked dense 2D to dense 3D. A 3D U-Net is trained to learn 3D affinities from stacked 2D LSDs using synthetic labels. **c,** Combined 2D to 3D inference pipeline. Sections from a 3D image volume are used as input to the trained 2D U-Net to generate stacked 2D LSDs. The trained 3D U-Net infers 3D affinities from the stacked 2D LSDs. A 3D segmentation is generated from the 3D affinities using standard post-processing techniques.

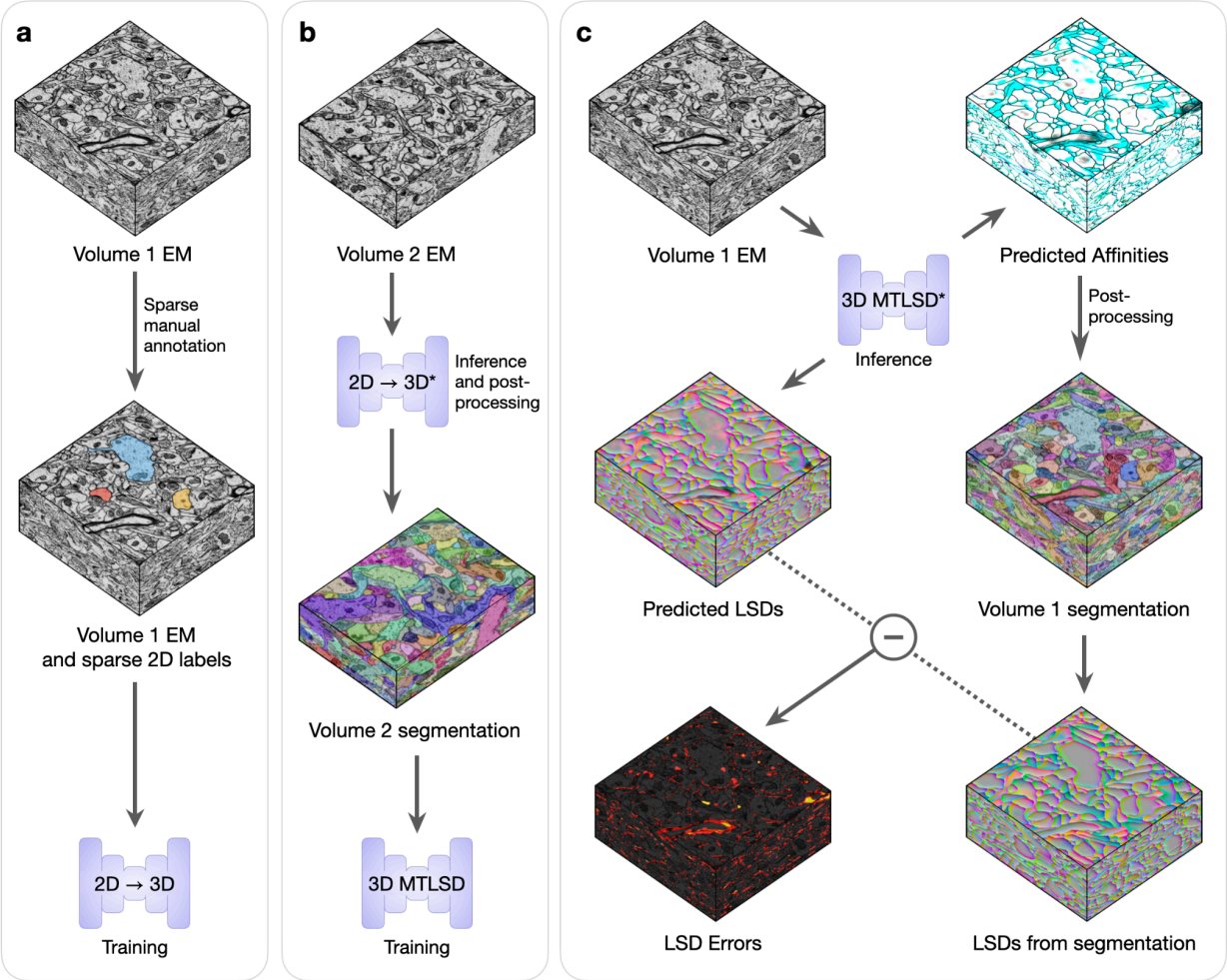


Fig. 2: Bootstrapping dense 3D segmentations. Volumes 1 and 2 are two EM volumes without any annotations. **a,** Sparse 2D ground-truth labels are manually created on Volume 1, which are used to train the 2D to 3D networks. **b,** A 3D segmentation of Volume 2 is generated using the trained 2D to 3D networks (denoted by *) and standard post-processing. The resulting 3D segmentation is used as pseudo ground-truth training data, without any masking or proofreading, for an untrained 3D MTLSD network. **c,** The trained 3D MTLSD network infers 3D affinities and LSDs on Volume 1. Post-processing is applied to the 3D affinities to generate a 3D segmentation on Volume 1, from which LSDs are computed. The element-wise difference between the model's LSDs and the computed segmentation LSDs generates a heat map of errors. These errors can be thresholded to obtain a mask of high-error regions, which can be used for filtering of segmentations and targeted refining during subsequent training.

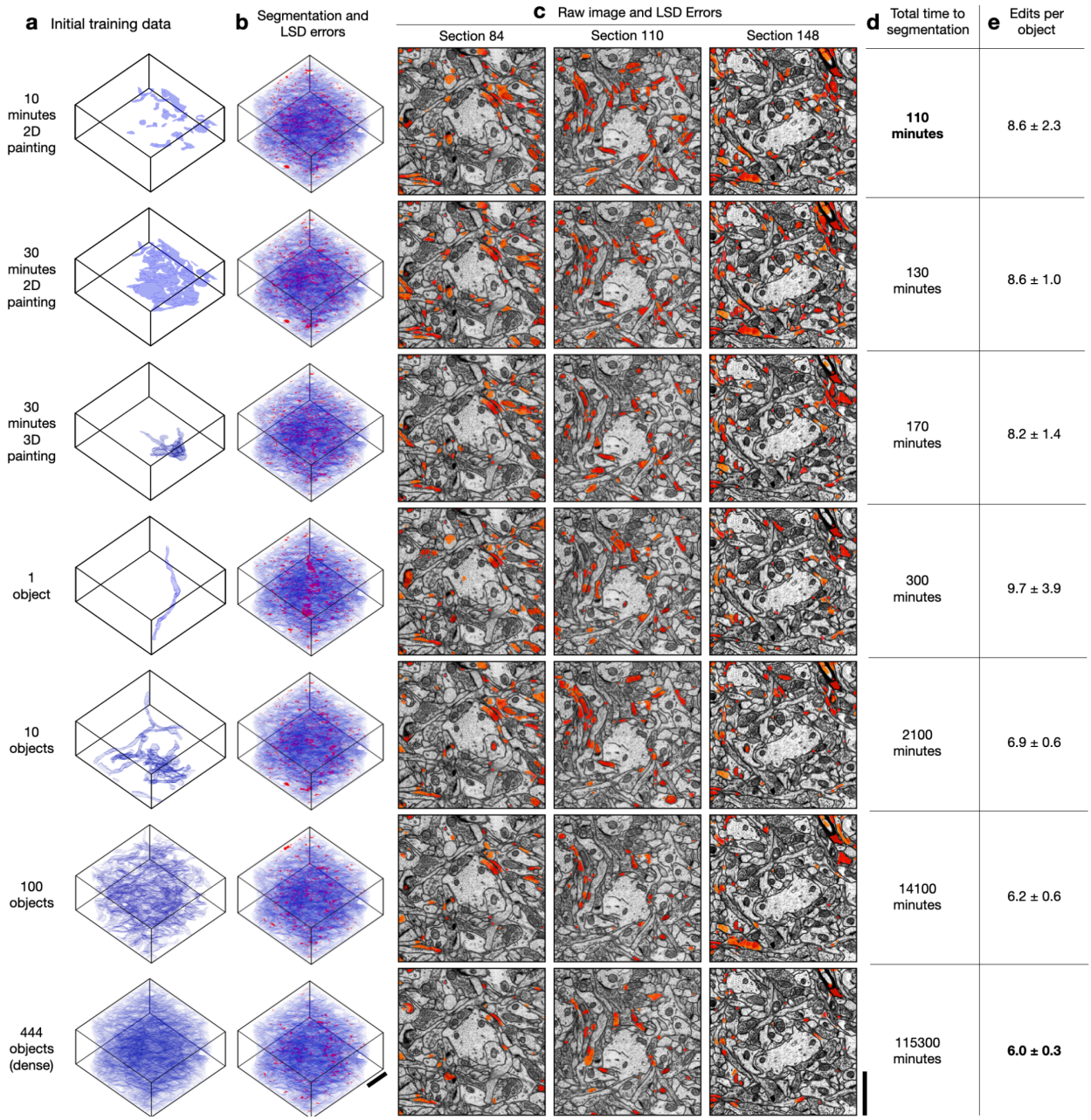


Fig. 3: Example from bootstrapped models for HARRIS-15. a, The first column shows different amounts of initial ground-truth training data used to generate a dense 3D segmentation of the test volume. **b,** Second column shows bootstrapped segmentations in agreement with ground-truth manual annotations (blue) and the LSD split/merge errors (red). The errors visualized are computed as the element-wise difference between the LSDs computed from ground-truth manual annotations and the LSDs computed from the bootstrapped segmentations. **c,** The third column shows a selection of 2D sections of the LSD errors overlaid on raw EM images with red areas showing examples of split or merge errors on that section. **d,** The fourth column shows the total time required to obtain a 3D segmentation starting from no annotations through bootstrapping and is a sum of the times for manual annotation, training 2D->3D, training 3D MTLSD, and inference and post-processing. **e,** The fifth column provides the average number (± standard deviation) of split and merge edits required for each object to match the dense ground-truth skeletons. Scale bars are 1.5 **µ**m.

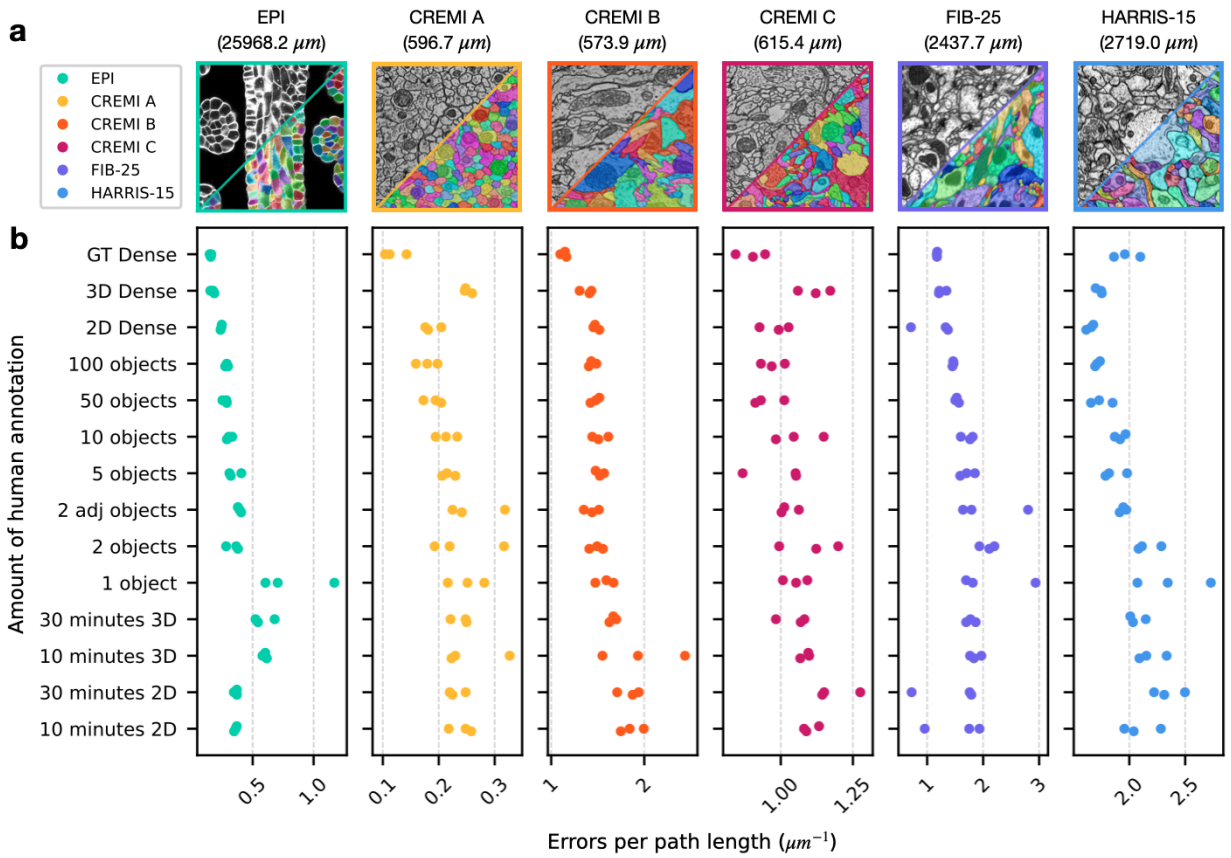


Fig. 4: Quantitative results of 3D segmentations bootstrapped from sparse annotations. a, Example images and ground-truth labels from each dataset and color-coding key. The total path length of the dense ground-truth skeletons in each training volume is indicated in parenthesis. **b,** Number of split and merge errors needing correction per skeleton path length to match the dense ground-truth skeletons. Lower scores are better. Each dataset had three separate tests for each amount of annotation (illustrated as three dots of the same color per row). Dense segmentations of a separate test volume (pseudo ground-truth) were first created by 2D->3D models trained on different amounts of initial ground truth annotations, which are sorted from bottom to top of the y-axis in order of increasing human annotation effort. Scores were then computed by comparing ground-truth annotations to segmentations produced by a 3D model trained on the pseudo ground-truth segmentation. 2D (or 3D) Dense refers to training the initial 2D->3D (or 3D) model on all available 2D (or 3D) annotations to generate pseudo ground-truth. GT Dense refers to directly training a 3D model on ground-truth annotations of the test volume to generate the segmentation of the initial training volume.

Supplementary Material

A. Sparsities

We define a *sparsity* as an amount of manual annotations. This sparse amount of manual annotations is used to train networks with a weighted loss function to infer dense predictions.

We sought to explore how sparse manual annotations can be in this context of neuron segmentation in electron microscopy (EM). Our objective was to identify efficient strategies for generating ground truth labels that provide sufficient information for the network's learning, while minimizing the associated human annotation costs.

We explored a total of 8 sparsities at the object-level:

- 1 object
- 2 objects
- 2 adjacent objects
- 5 objects
- 10 objects
- 50 objects
- 100 objects
- All objects (dense)

For each of the above object-level sparsities, we included three additional disk sparsities, resulting in a total of 32 sparsities (Supp. Fig. 1). We included disk sparsities based on considerations from the local shape descriptors (LSDs); the regions thought to be most crucial for learning LSDs are concentrated around object boundaries and entirely within objects[1]. Under this assumption, sparsity could theoretically be represented by deliberately positioned incomplete objects (such as paint strokes). For example, placing a stroke on one side of an object, another stroke on the opposite side, and a stroke in the center could contain much of the necessary information to instruct a network about the presence of a smooth gradient inside an object and sharp transitions across boundaries. Unlabeled areas could then be considered unknown using a weighted loss function. Hence, for a disk sparsity of N disks, we selected N points within the field of view (FOV) and drew circles with a sufficient radius intersecting the labels.

The intersecting circle cuts off the labels in disk sparsities, leading to incorrect ground truth LSDs towards the center of objects where there should be a sharp gradient instead of a smooth or small gradient. We expected this discrepancy to adversely impact predictions and result in circular artifacts or false boundaries. However, we were surprised to observe that the network still predicted these regions correctly. This could be attributed to several factors:

Firstly, we randomly sampled enough locations in batches that were deemed correct. An ablation study could involve limiting these circles to always include incorrect boundaries. Secondly, the network might become confused in these regions due to conflicting signals from the "correct" regions. Consequently, the network regresses to predict approximately 0.5 in these areas, which happens to map to gray in RGB space, akin to the intentional design of the original LSDs.

B. Experiment and grid search

The experimental procedure in Fig. 2 is described here in more detail. For every dataset, sparsity, and repetition:

1. The 2D U-Net of the 2D->3D method was trained on the sparse 2D annotations of Volume 1. The three variations of the 2D U-Nets explored differ by output: 2D affinities, 2D LSDs, or both (2D MTLSD).
2. Inference was done with different iterations of the trained 2D->3D models to generate predictions of 3D affinities of Volume 2.
    a. Stacked 2D predictions are made with an iteration of the trained 2D U-Net. The predictions tested were: 2D affinities, 2D LSDs, 2D affinities (from MTLSD), 2D LSDs (from MTLSD). (4 variations)
    b. 3D affinities of Volume 2 are made with an iteration of the trained 3D U-Net of the 2D->3D method. All four strategies to generate synthetic 3D labels during training were explored. (4 variations)
3. Post-processing was done in a grid-fashion to generate 3D segmentations of Volume 2 from all different combinations of model iterations, network variations, and post-processing parameters. (Supp. Table 3a)
4. All the generated segmentations of Volume 2 were evaluated for accuracy (Supp. Fig. X-Y.). The best segmentation was designated as the pseudo ground-truth training data for the untrained 3D MTLSD model without any proofreading.
5. The 3D MTLSD model was trained on the pseudo ground-truth training data of Volume 2.
6. Inference was done with different iterations of the trained 3D MTLSD model to generate predictions of 3D affinities of Volume 1.
7. Post-processing was done in a grid-fashion to generate 3D segmentations of Volume 1 from the different predictions. (Supp. Table 3b)
8. All the generated segmentations of Volume 1 were evaluated for accuracy. (Fig. 3, Supp. Fig.s 3-4). Total time to segmentation starting from sparse annotation was estimated (Fig. 4). The best segmentation in terms of accuracy was designated as the representative bootstrapped segmentation for the dataset, sparsity, and rep in question.

C. Evaluation metrics

We automatically generated ground-truth skeletons for evaluation from ground-truth labels with the following steps. We used a watershed algorithm on computed ground-truth affinities to

generate an oversegmentation (resulting in supervoxels). Each supervoxel center of mass was stored as a node with position coordinates in a region adjacency graph. For each ground-truth mask, we computed the minimum spanning-tree of the nodes using the physical distance between nodes as the weight.

We compare the accuracy of bootstrapped segmentations using the normalized, variation of information (VOI) metric and min-cut metric (MCM)[1–3]. VOI is the voxel-based measure of similarity between a segmentation and ground-truth labels. VOI reports the amount of split and merge errors. We note that VOI can be sensitive to slight differences in boundaries. At the same time, small topological changes might go unnoticed, which is especially problematic in fine neuropil. Nevertheless, VOI can be a good proxy for segmentation quality[1].

Quantifying segmentation accuracy in terms of proofreading effort required to correct it is more interpretable and relevant to optimize. False splits require only one interaction to merge. False merges can also be fixed with a few interactions[1,4,5]. The MCM measures the total number of split and merge edit operations needed to make a segmentation agree with ground-truth skeletons. We report the total edits per object, total edits per path length, total splits and total merges per object.

D. Network architectures

All networks were trained using gunpowder (https://funkelab.github.io/gunpowder/) and PyTorch, using the U-Net architecture implemented in (github.com/funkelab/funlib.learn.torch). Code for LSDs is available at (github.com/funkelab/lsd) and code for example 2D->3D networks and 3D MTLSD networks are available at (github.com/ucsdmanorlab/bootstrapper).

The 3D MTLSD network used for bootstrapping was a 3D UNet with two separate decoder heads for the two learning tasks: affinities and LSDs. The MTLSD U-Net consisted of three layers with downsampling factors [1,2,2]. The bottleneck and the adjacent layers have 2D convolution kernels. Thirteen initial feature maps were used with a multiplication factor of 6 between layers. The resulting data was further convolved and passed through a sigmoid activation to get from 13 output feature maps from each decoder head to 3 (3D affinities) and 10 (3D LSDs).

The 2D U-Nets used in the 2D->3D method consisted of three layers and were downsampled by a factor of [2,2] in all layers. Twelve initial feature maps were used and features were multiplied by a factor of 6 between layers. The resulting data was further convolved and passed through a sigmoid activation to get from 12 output feature maps to either 2 (2D affinities), 6 (2D LSDs) or 8 feature maps (2D MTLSD).

The lightweight 3D U-Nets in the 2D->3D method consisted of two layers and were downsampled by a factor of [1,2,2]. The convolution kernel sizes for the first downsampling and last upsampling layers are [2,3,3] and [1,3,3] for the other layers. 5 initial feature maps were used and features were multiplied by a factor of 5 between layers. The resulting data was further convolved and passed through a sigmoid activation to get from 5 output feature maps to 3 (3D affinities). All networks used a mean-squared error loss, minimized with an Adam optimizer.

E. Training pipelines

For all sparse 2D and 3D networks and the 3D MTLSD network used for bootstrapping, each training batch was randomly picked from the available sections or volume. For each batch, the raw data was first normalized and padded with zeros. Labels were padded with the maximum padding required to contain at least 0.01% (10% for 3D MTLSD, since there are more pseudo GT labels available) of labeled ground-truth data within the image assuming a worst case rotation of 45 degrees. Data was randomly sampled and augmented with elastic transformations, random mirrors and transposes, gaussian blur, and intensities (see Supplementary Table 2 for augmentation hyper-parameters used for all networks). For the networks with affinities as an output, a scale array was created to balance loss between target affinity labels.

For the lightweight 3D networks in the 2D->3D method, each training batch begins as a 3D array of zeros. Synthetic 3D labels are randomly grown with the strategies listed below. Labels were then augmented with elastic transformations and random mirrors and transposes, after which they were used to simulate stacked 2D affinities or LSDs. The stacked 2D affinities or LSDs were then augmented with random noise, intensities and gaussian blur to simulate realistic stacked 2D predictions. Finally, target 3D affinities were computed and a scale array was created to balance loss between class labels.

Synthetic 3D Labels Generation

A. N many randomly chosen voxels in the array of zeros get set to 1, where N is a random integer between 25 and 50. The speckled array is relabeled such that each labeled voxel has a unique integer ID. The labels are grown outward by D pixels without overlapping other labels, where D is another random integer between 25 and 40. The ID of the background label is bumped to a non-zero and unique integer ID.
B. Similar to A, but the speckled binary array is dilated section-wise with the binary structure of a 2x2 square or a disk binary structure with a random radius between 1 and 5. The binary array is then labeled such that each foreground instance has a unique integer ID. The uniquely labeled objects are then expanded into unoccupied spaces using a distance transform of the background.
C. A gaussian filter with sigma=5 is applied to an array of random float values to obtain peaks. The peaks are accentuated with a maximum filter with a size of 10 pixels and

maxima are identified as seeds. Watershed is then applied to the peaks to grow labels from the seeds.

D. An equal mix of the above three strategies.

Dataset Name	Imaging modality	Content	Resolution (nm/px, ZYX)	Size (pixels, ZYX)	Number of objects
HARRIS-15[6]	TEM	*Rattus* hippocampal neuropil	(50, 2, 2), downscaled to (50, 8, 8)	(101,900,927)	444
				(70,674,504)	181
FIB-25[7]	FIBSEM	*Drosophila* optic medulla neuropil	(8, 8, 8)	(520,520,520)	1690
				(520,520,520)	2031
CREMI-C[8]	SSTEM	*Drosophila* calyx neuropil	(40, 4, 4) Downscaled to (40, 8, 8)	(62,625,625)	546
				(63,625,625)	647
CREMI-B[8]		*Drosophila* axon tract		(62,625,625)	450
				(63,625,625)	581
CREMI-A[8]		*Drosophila* axon tract		(63,625,625)	333
				(62,625,625)	312
EPI[9]	LM	*Arabidopsis* epithelial cells	(235, 75, 75)	(318,960,953)	3333
				(248,791,667)	1053

Supp. Table 1: Overview of datasets. Dataset rows show Volume 1 above Volume 2. A size filter of 500 pixels was applied to the labels before counting objects.

Parameter	Value
Input feature maps	12
Layer feature map scale	6
Downsampling factors	[[2, 2], [2, 2], [2, 2]]

Input shape	[196, 196]
Output shape	[104, 104]
Loss	Weighted MSE
Optimizer	Adam
Learning rate	0.5 × 10^-4
β1	0.9
β2	0.999
ε	1 × 10^-8
Iterations	20,000

Augmentation	Parameter	Value
Elastic	Control point spacing	(8, 8)
	Jitter Sigma	(2, 2)
	Subsample	4
Rotation	Axis	x, y
	Angle	in [0, 2π]
Mirror	Axes	x, y
Transpose	Axes	x, y
Noise	Mode	Gaussian
Intensity	Scale	in [0.9, 1.1]
	Shift	in [−0.1, 0.1]
Blur	Sigma	In [0.0, 1.5]

Supp. Table 2: Training parameters and augmentations of 2D networks of the 2D->3D method on HARRIS-15.

Parameter	Value
Input feature maps	5
Layer feature map scale	5
Downsampling factors	[[1, 2, 2], [1, 2, 2]]
Kernel sizes down	[[2, 3, 3], [2, 3, 3]], [[1, 3, 3], [1, 3, 3]], [[1, 3, 3], [1, 3, 3]]

Kernel sizes up	[[1, 3, 3], [1, 3, 3]], [[2, 3, 3], [2, 3, 3]],
Input shape	[10, 148, 148]
Output shape	[6, 108, 108]
Loss	Weighted MSE
Optimizer	Adam
Learning rate	0.5 × 10^-4
β1	0.9
β2	0.999
ε	1 × 10^-8
Iterations	20,000

Supp. Table 3: Network and training parameters of 3D networks in the 2D->3D method on HARRIS-15.

Parameter	**Values**	**Number of variations**
Synthetic 3D labels generation	A, B, C, D	4
3D affinities iteration	{5000, 10000, 15000, 20000} (2D U-Net) x {5000, 10000, 15000, 20000} {3D U-Net)	16
Rep	rep_1, rep_2, rep_3	3
Sparsity	10min_paint_{2d,3d} + paint_{2d, 3d} + {disk_{0,1,2,3}} x obj_{001,002,002a,005,010,100,dense}} + 3D_dense	33
Watershed minimum seed distance	10	1
Hierarchical merge function	“mean”, “hist_quant_50”, “hist_quant_75”	3
Total parameter grid size		19008

Supp. Table 4: Post-processing parameter grid explored for generating 2D->3D segmentations on HARRIS-15.

Parameter	Value
Input feature maps	13
Layer fmap scale	6
Downsampling factors	[[1, 2, 2], [1, 2, 2], [1, 2, 2]]
Kernel sizes down	[[3, 3, 3], [3, 3, 3]], [[3, 3, 3], [3, 3, 3]], [[1, 3, 3], [1, 3, 3]], [[1, 3, 3], [1, 3, 3]]
Kernel sizes up	[[1, 3, 3], [1, 3, 3]], [[3, 3, 3], [3, 3, 3]], [[3, 3, 3], [3, 3, 3]]
Input shape	[20, 196, 196]
Output shape	[4, 104, 104]
Loss	Weighted MSE
Optimizer	Adam
Learning rate	0.5 × 10^-4
β1	0.9
β2	0.999
ε	1 × 10^-8
Iterations	50,000

Augmentation	Parameter	Value
Elastic	Control point spacing	(8, 50, 50)
	Jitter Sigma	(0, 2, 2)
	Subsample	4
	Scale interval	(0.75, 1.25)
Rotation	Axis	x, y
	Angle	in [0, 2π]
Mirror	Axes	x, y, z
Transpose	Axes	x, y
Noise	Mode	Gaussian
Intensity	Scale	in [0.9, 1.1]
	Shift	in [−0.1, 0.1]
Blur	Sigma	In [0.0, 1.5]

Supp. Table 5: Training parameters and augmentations of 3D networks on HARRIS-15.

a

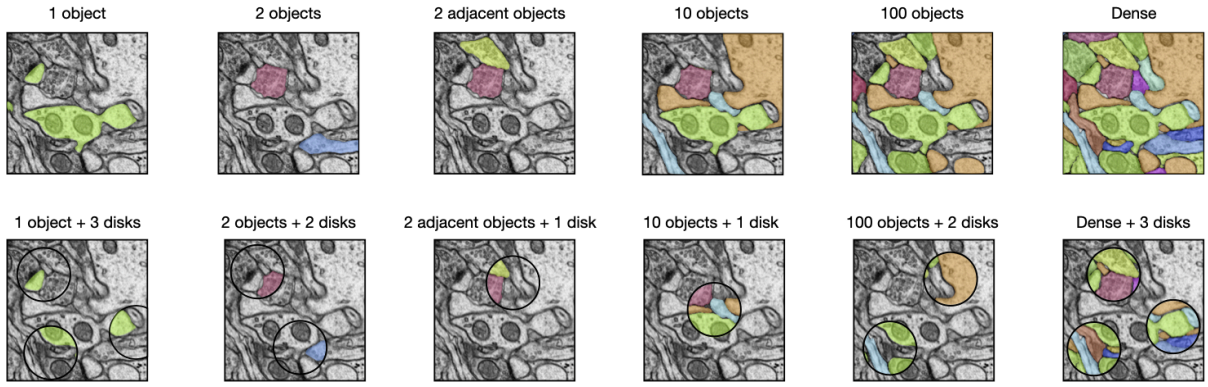
Parameter	Values	Number of variations
Predicted affinities iteration	20000, 35000, 50000	3
Rep	rep_1, rep_2, rep_3	3
Pseudo GT network	Affinities, LSDs, Affinities from MTLSD, LSDs from MTLSD	4
Pseudo GT Sparsities	10min_paint_{2d,3d} + paint_{2d, 3d} + obj_{001,002,002a,005,010,100,2D_dense, 3D_dense}}	13
Normalize affinities	False	1
Watershed minimum seed distance	10	1
Watershed boundary mask	True	1
Hierarchical merge function	“mean”	1
Total parameter grid size		468

b

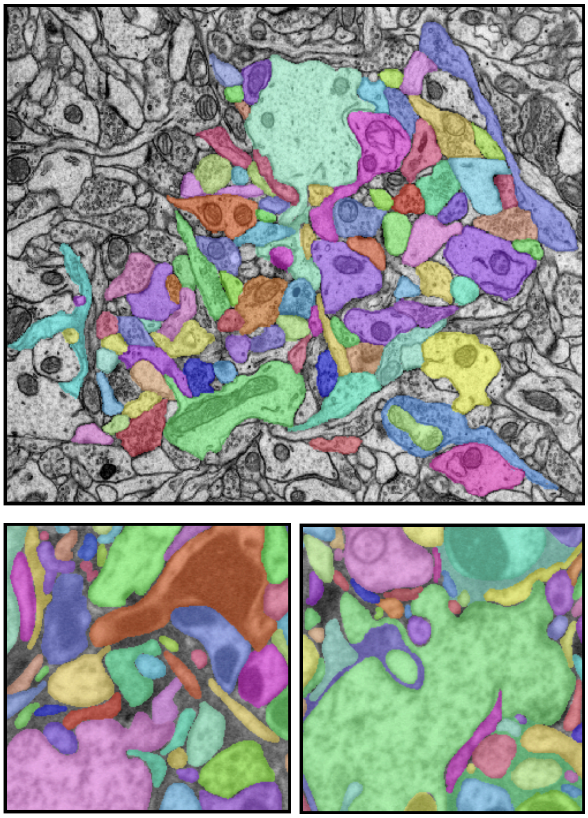
Parameter	Values	Number of variations
Predicted affinities iteration	20000, 35000, 50000	3
Rep	rep_1, rep_2, rep_3	3
Pseudo GT network	Affinities, LSDs, Affinities from MTLSD, LSDs from MTLSD	4
Pseudo GT Sparsities	10min_paint_{2d,3d} + paint_{2d, 3d} + obj_{001,002,002a,005,010,100,2D_dense, 3D_dense, GT_dense}}	13

Normalize affinities	False, True	2
Watershed minimum seed distance	5, 10, 15, 20	4
Watershed boundary mask	True, False	2
Hierarchical merge function	'hist_quant_10', 'hist_quant_25', 'hist_quant_50', 'hist_quant_75', 'hist_quant_90', 'mean'	6
Total parameter grid size		44928

Supp. Table 6: Post-processing parameter grid explored for generating bootstrapped segmentations on HARRIS-15. a, VOI-only grid. **b,** VOI and MCM included in the grid.

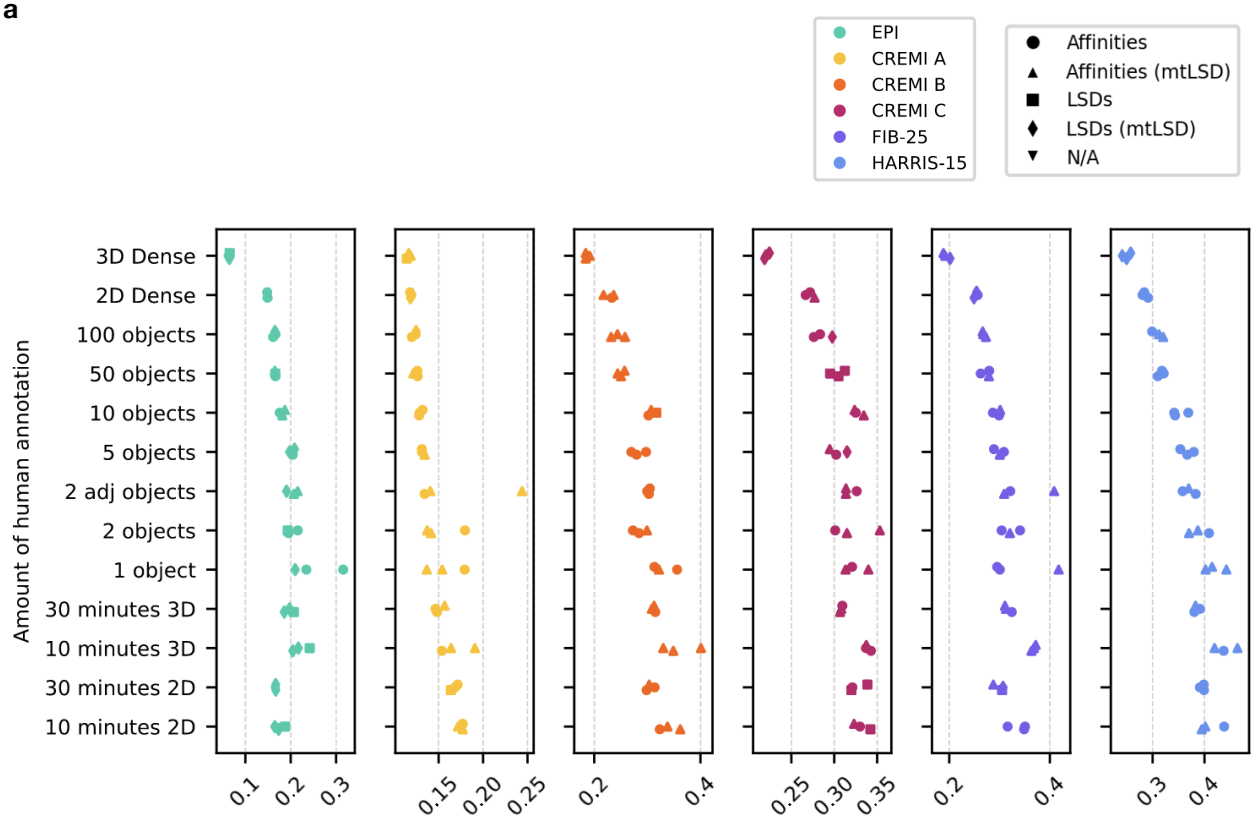


Supp. Fig. 1 | Example simulated sparsity levels. Example HARRIS-15 image and ground-truth labels in training batches for different simulated sparsity levels which involve object-level ablations and disk-selections.

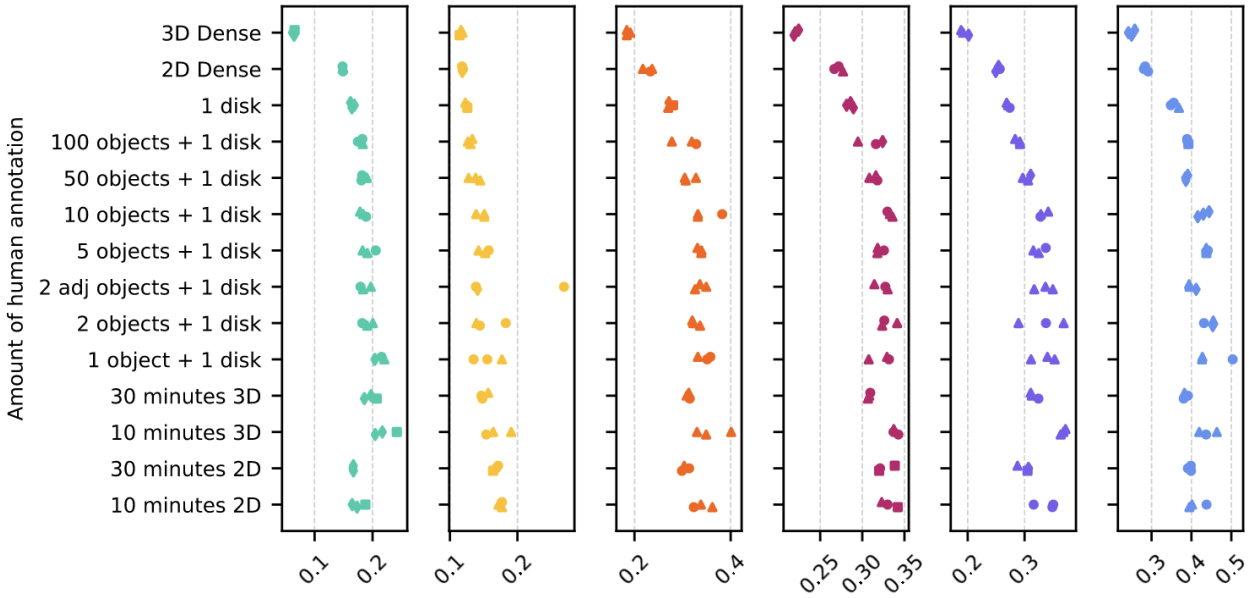


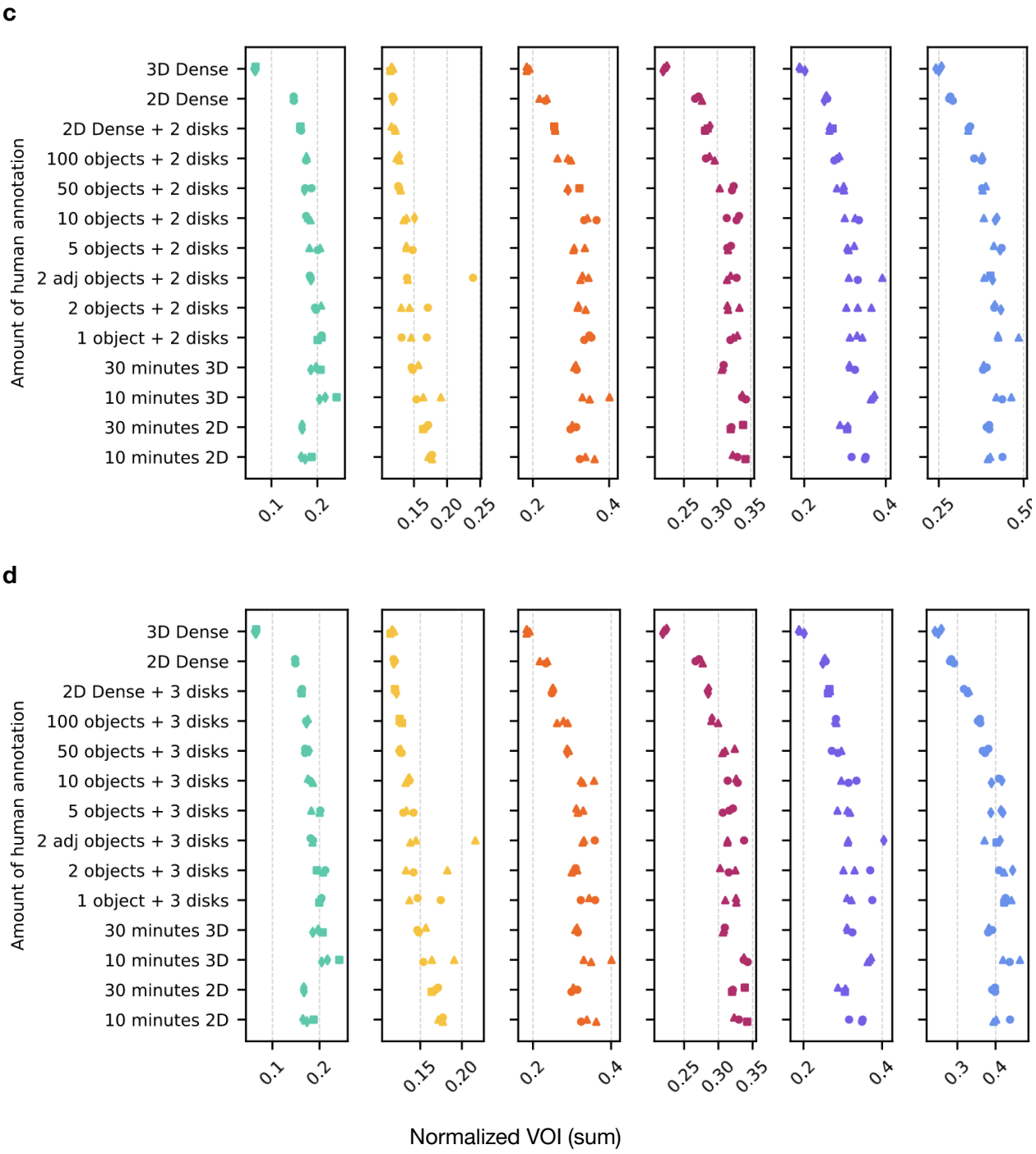
Supp. Fig. 2 | Example non-expert 2D annotations. Example image and non-expert annotations for HARRIS-15 (top) and FIB-25 (bottom) made in 30 minutes.

a

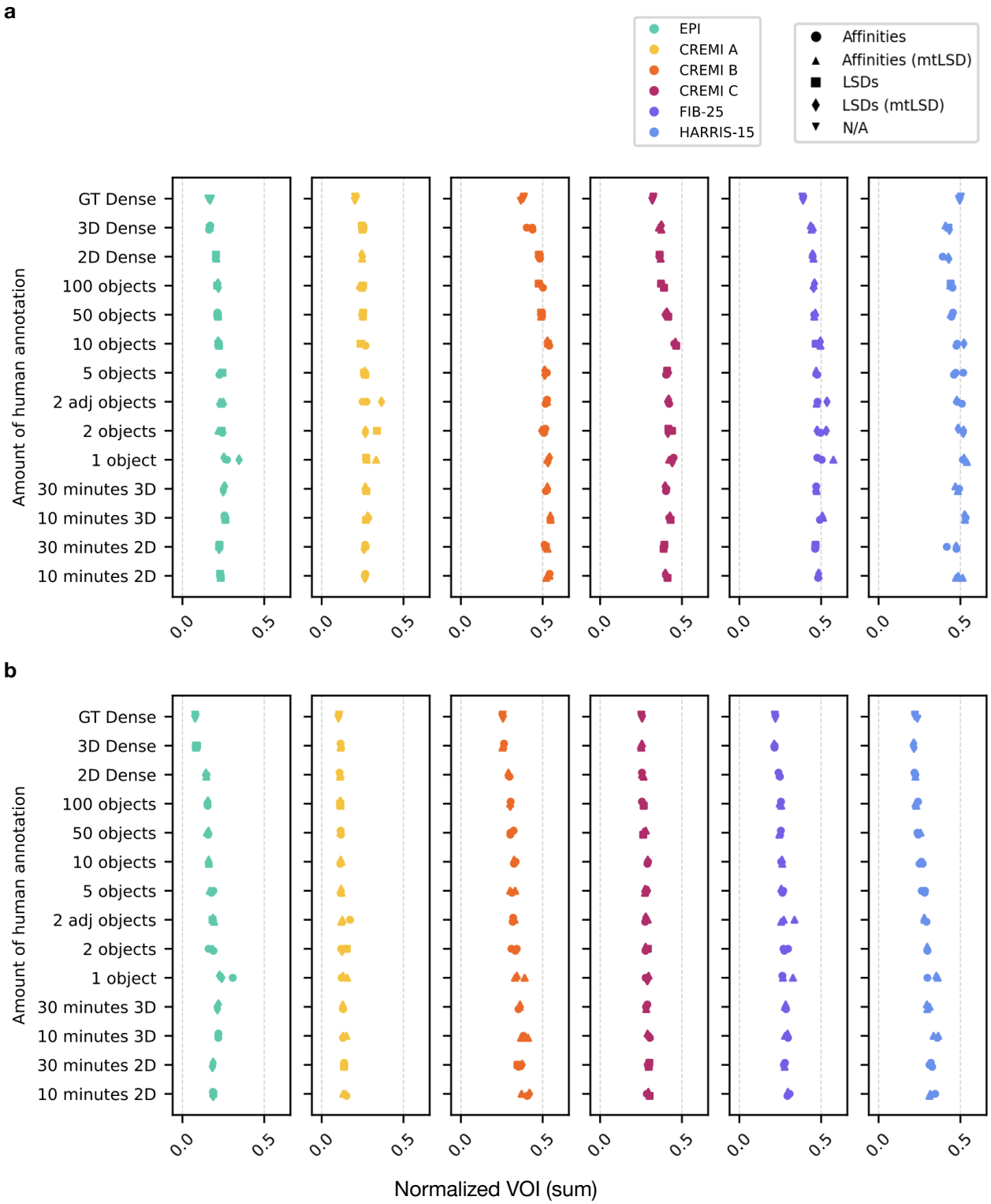

EPI
CREMI A
CREMI B
CREMI C
FIB-25
HARRIS-15
Affinities
Affinities (mtLSD)
LSDs
LSDs (mtLSD)
N/A
Amount of human annotation
3D Dense
2D Dense
100 objects
50 objects
10 objects
5 objects
2 adj objects
2 objects
1 object
30 minutes 3D
10 minutes 3D
30 minutes 2D
10 minutes 2D
0.1
0.2
0.3
0.15
0.20
0.25
0.2
0.4
0.25
0.30
0.35
0.2
0.4
0.3
0.4

b


Amount of human annotation
3D Dense
2D Dense
1 disk
100 objects + 1 disk
50 objects + 1 disk
10 objects + 1 disk
5 objects + 1 disk
2 adj objects + 1 disk
2 objects + 1 disk
1 object + 1 disk
30 minutes 3D
10 minutes 3D
30 minutes 2D
10 minutes 2D
0.1
0.2
0.1
0.2
0.2
0.4
0.25
0.30
0.35
0.2
0.3
0.3
0.4
0.5



Supp. Fig. 3: Quantitative results of 2D->3D models. Normalized Variation of Information (VOI) sum scores, lower scores are better. Scores were computed by comparing ground-truth labels to segmentations produced by the 2D->3D method. Plot includes best scores from parameter grid-searches, for each dataset and repetition versus the sparse paintings and **a.** object-level sparsities, **b, c,** and **d** with 1, 2, and 3 intersecting disk(s) per training batch, respectively. Marker shapes indicate the best performing 2D->3D or 3D strategy. Dense refers to un-ablated labels. 2D (or 3D) Dense refers to the segmentation generated by the 2D->3D method (or a 3D model) after training on un-ablated 2D (or 3D) labels.

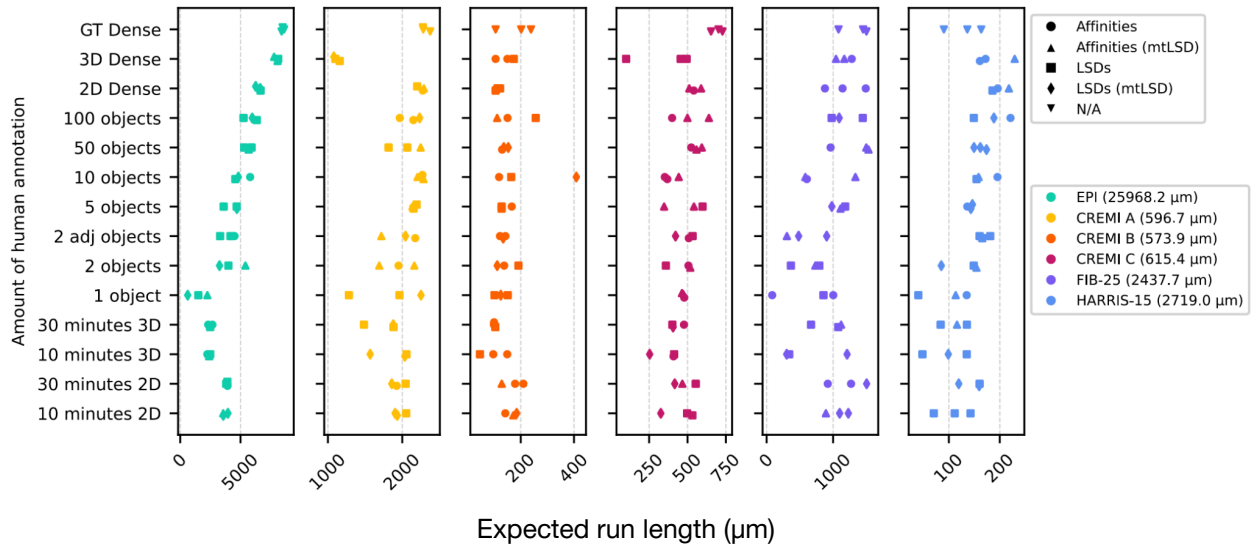


Supp. Fig. 4: Effect of total parameter grid size on quantitative results.

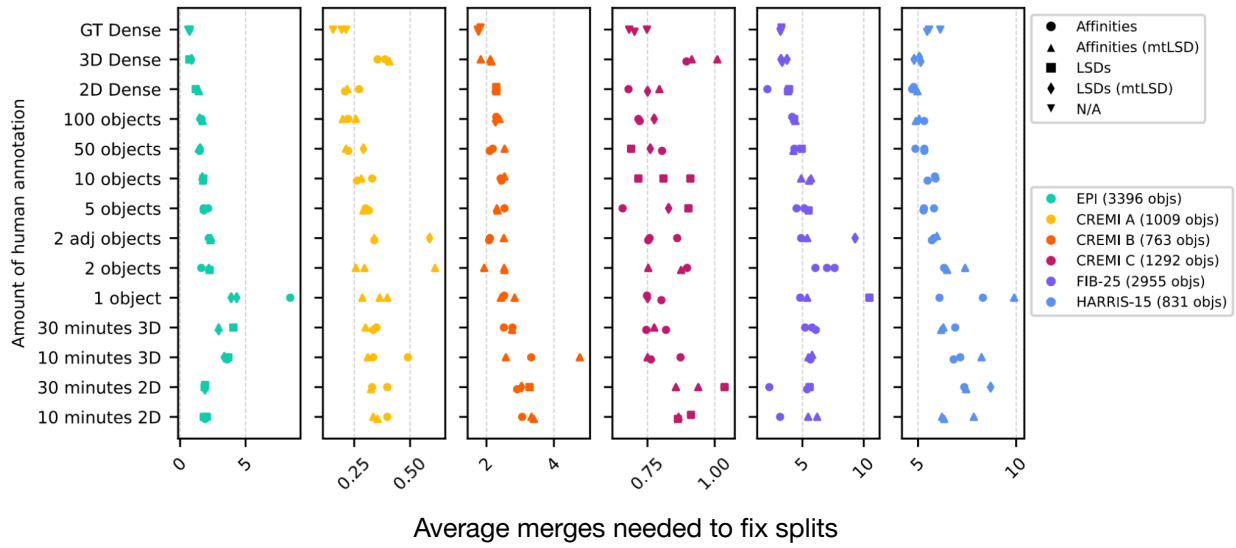
Lower scores are better. Marker shapes indicate the best performing 2D->3D strategy. Normalized Variation of Information (VOI) sum scores of bootstrapped 3D models with parameter grid size per dataset **a,** 44928 runs and **b,** 468 runs. See Supp. Table 6 for parameters used.

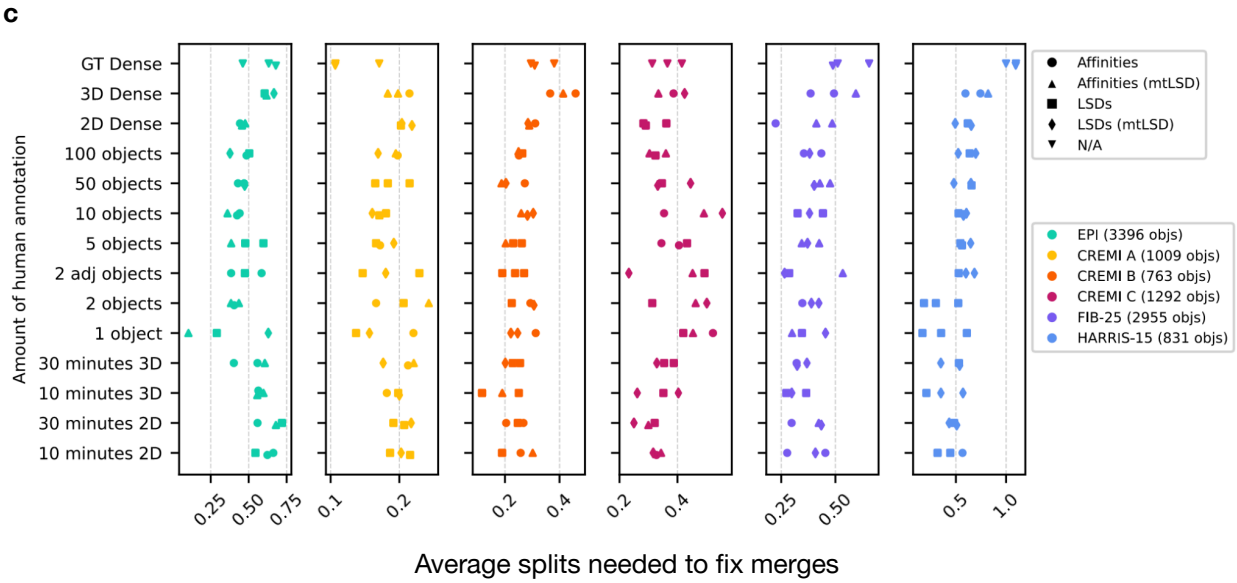
---

a



b





Supp. Fig. 5 | Quantitative results of bootstrapped 3D models. Lower scores are better. Marker shapes indicate the best performing 2D->3D strategy. **a,** Expected run length[10]. **b,** Average merges needed to fix splits. **c,** Average splits needed to fix merges. Each dataset had three separate tests for each amount of annotation (illustrated as three dots of the same color per row). Dense segmentations of a separate test volume (pseudo ground-truth) were first created by 2D->3D models trained on different amounts of initial ground truth annotations, which are sorted from bottom to top of the y-axis in order of increasing human annotation effort. Scores were then computed by comparing ground-truth annotations to segmentations produced by a 3D model trained on the pseudo ground-truth segmentation. 2D (or 3D) Dense refers to training the initial 2D->3D (or 3D) model on all available 2D (or 3D) annotations to generate pseudo ground-truth. GT Dense refers to directly training a 3D model on ground-truth annotations of the test volume to generate the segmentation of the initial training volume.

References

1. Sheridan, A. *et al.* Local shape descriptors for neuron segmentation. *Nat. Methods* **20**, 295–303 (2023).
2. Meilă, M. Comparing clusterings—an information based distance. *J. Multivar. Anal.* **98**, 873–895 (2007).
3. Vinh, N. X., Epps, J. & Bailey, J. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *J. Mach. Learn. Res.* **11**, 2837–2854 (2010).
4. Zhao, T., Olbris, D. J., Yu, Y. & Plaza, S. M. NeuTu: Software for Collaborative, Large-Scale,

Segmentation-Based Connectome Reconstruction. *Front. Neural Circuits* **12**, 101 (2018).

5. Dorkenwald, S. *et al.* FlyWire: online community for whole-brain connectomics. *Nat. Methods* **19**, 119–128 (2022).
6. Harris, K. M. *et al.* A resource from 3D electron microscopy of hippocampal neuropil for user training and tool development. *Sci. Data* **2**, 150046 (2015).
7. Takemura, S. *et al.* Synaptic circuits and their variations within different columns in the visual system of Drosophila. *Proc. Natl. Acad. Sci.* **112**, 13711–13716 (2015).
8. CREMI. https://cremi.org/.
9. Wolny, A. *et al.* Accurate and versatile 3D segmentation of plant tissues at cellular resolution. *eLife* **9**, e57613 (2020).
10. Januszewski, M. *et al.* High-precision automated reconstruction of neurons with flood-filling networks. *Nat. Methods* **15**, 605–610 (2018).

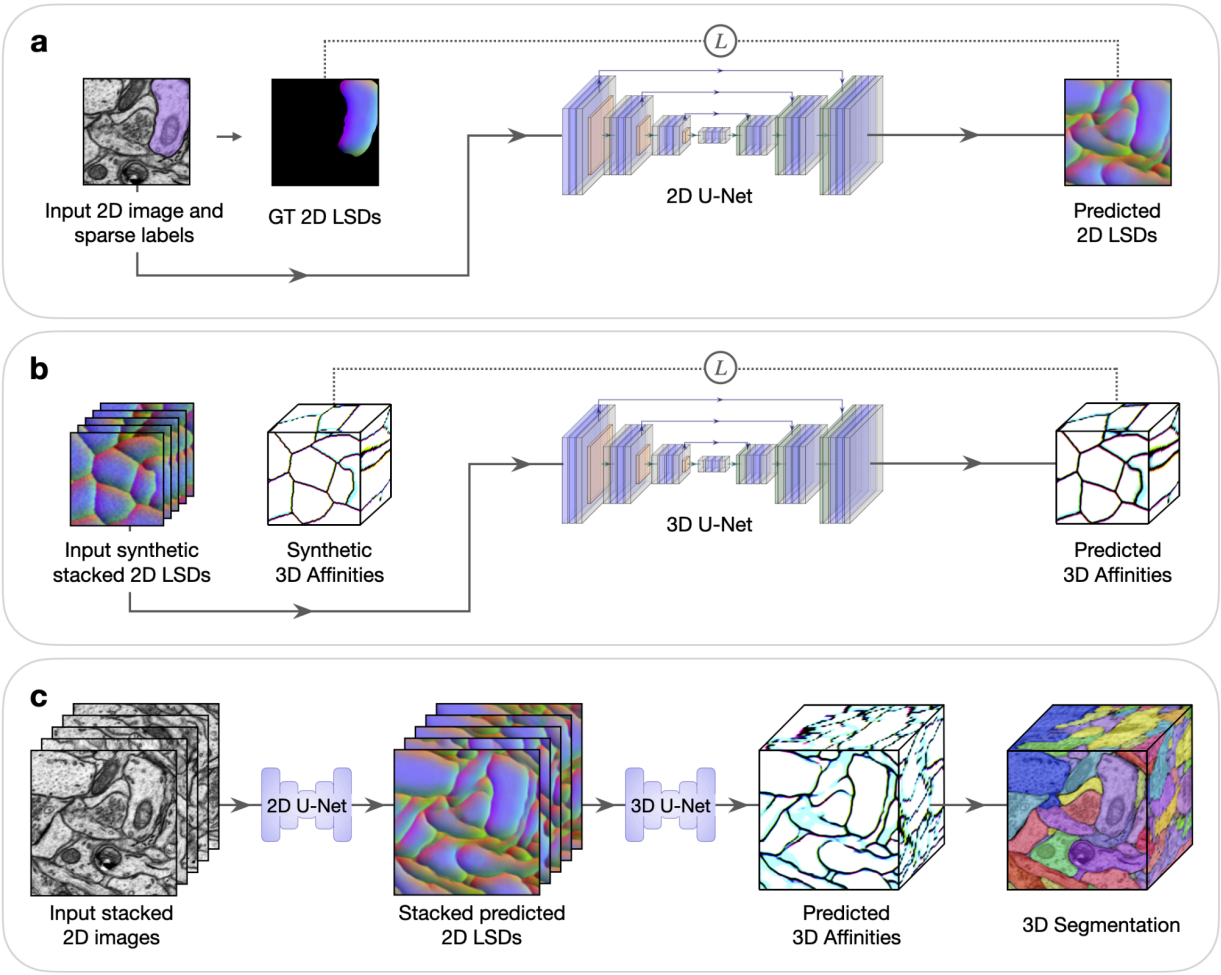


Fig. 1: Method to generate dense 3D instance segmentations from sparse 2D training data. Five example sections are shown in stacked 2D images or LSDs to simplify visualization. LSDs show the first three components as RGB-channels. All networks use the weighted mean-squared-error (MSE) loss function for training, denoted by L. **a,** Sparse 2D to dense 2D. Input 2D images with sparse ground-truth labels are used to train a 2D U-Net to learn dense 2D LSDs. Background regions of the ground-truth 2D LSDs are not used in the computation of the loss during training. **b,** Stacked dense 2D to dense 3D. A 3D U-Net is trained to learn 3D affinities from stacked 2D LSDs using synthetic labels. **c,** Combined 2D to 3D inference pipeline. Sections from a 3D image volume are used as input to the trained 2D U-Net to generate stacked 2D LSDs. The trained 3D U-Net infers 3D affinities from the stacked 2D LSDs. A 3D segmentation is generated from the 3D affinities using standard post-processing techniques.

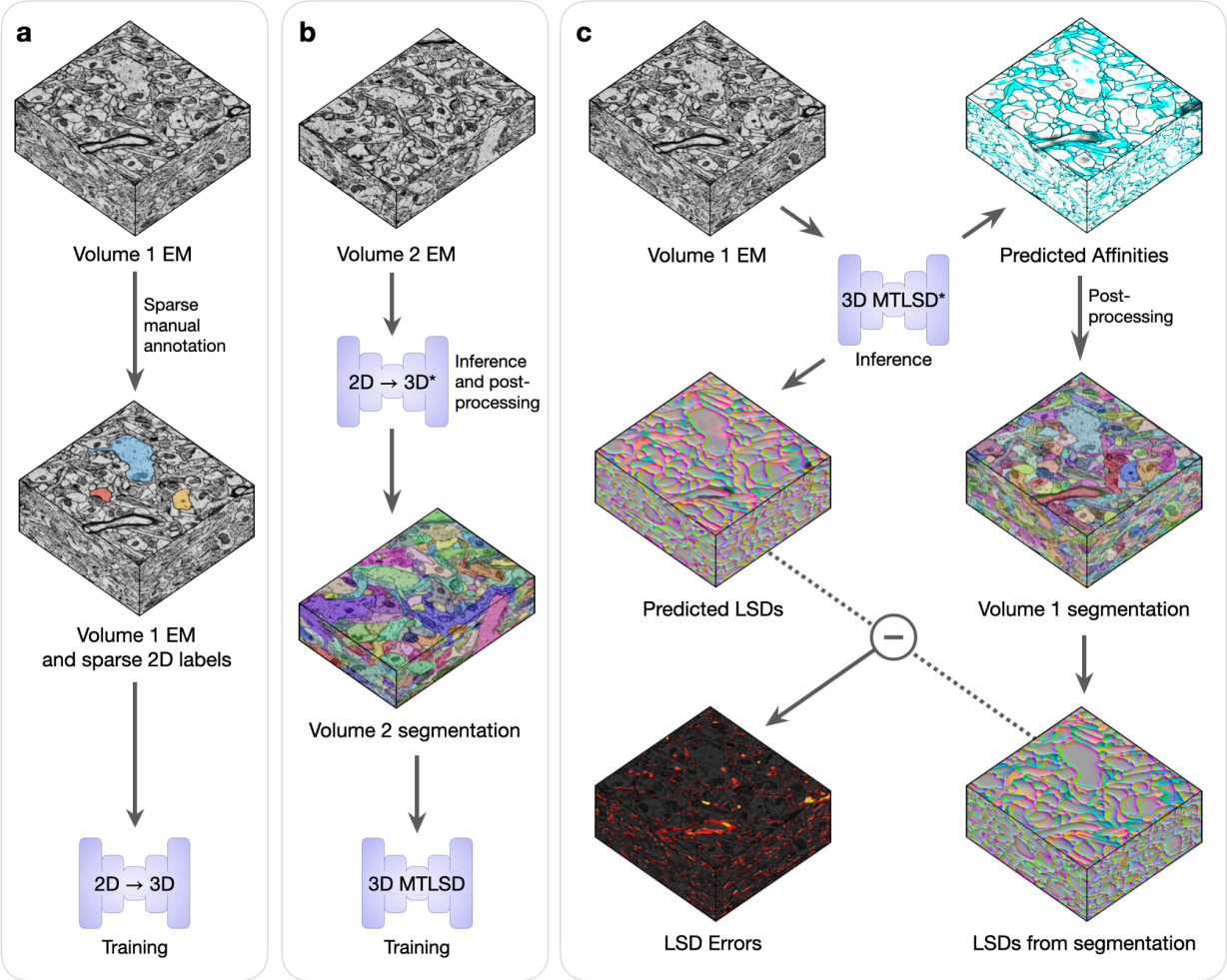


Fig. 2: Bootstrapping dense 3D segmentations. Volumes 1 and 2 are two EM volumes without any annotations. **a,** Sparse 2D ground-truth labels are manually created on Volume 1, which are used to train the 2D to 3D networks. **b,** A 3D segmentation of Volume 2 is generated using the trained 2D to 3D networks (denoted by *) and standard post-processing. The resulting 3D segmentation is used as pseudo ground-truth training data, without any masking or proofreading, for an untrained 3D MTLSD network. **c,** The trained 3D MTLSD network infers 3D affinities and LSDs on Volume 1. Post-processing is applied to the 3D affinities to generate a 3D segmentation on Volume 1, from which LSDs are computed. The element-wise difference between the model's LSDs and the computed segmentation LSDs generates a heat map of errors. These errors can be thresholded to obtain a mask of high-error regions, which can be used for filtering of segmentations and targeted refining during subsequent training.

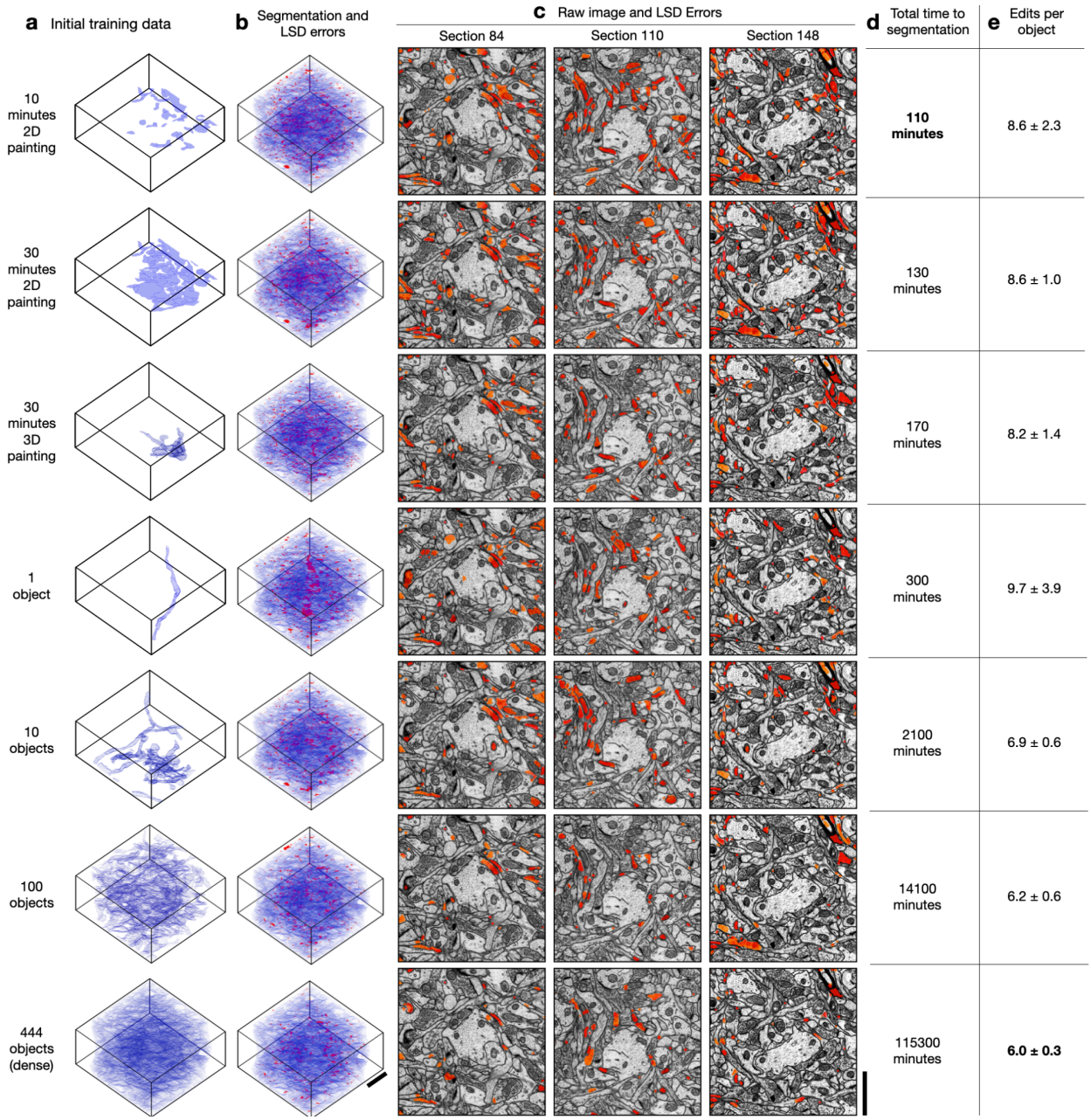


Fig. 3: Example from bootstrapped models for HARRIS-15. a, The first column shows different amounts of initial ground-truth training data used to generate a dense 3D segmentation of the test volume. **b,** Second column shows bootstrapped segmentations in agreement with ground-truth manual annotations (blue) and the LSD split/merge errors (red). The errors visualized are computed as the element-wise difference between the LSDs computed from ground-truth manual annotations and the LSDs computed from the bootstrapped segmentations. **c,** The third column shows a selection of 2D sections of the LSD errors overlaid on raw EM images with red areas showing examples of split or merge errors on that section. **d,** The fourth column shows the total time required to obtain a 3D segmentation starting from no annotations through bootstrapping and is a sum of the times for manual annotation, training 2D->3D, training 3D MTLSD, and inference and post-processing. **e,** The fifth column provides the average number (± standard deviation) of split and merge edits required for each object to match the dense ground-truth skeletons. Scale bars are 1.5 **µ**m.

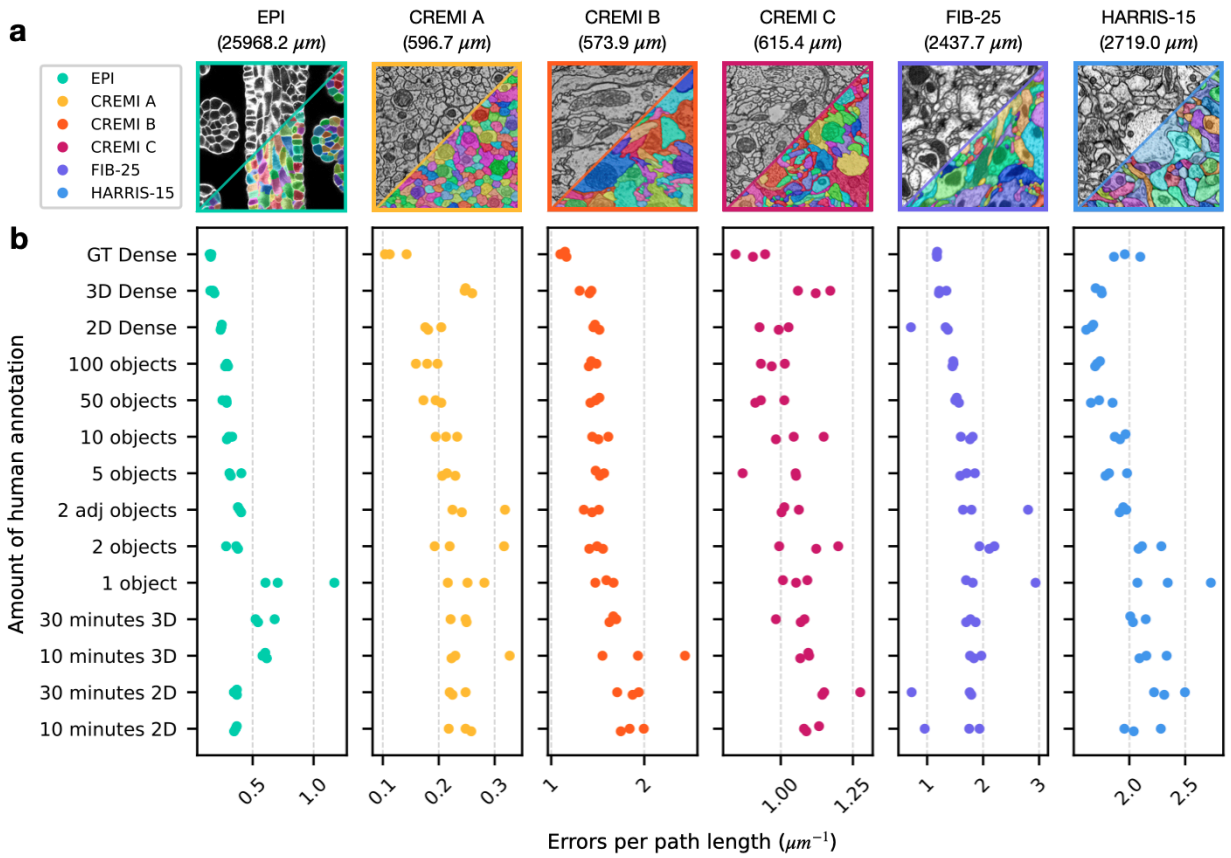


Fig. 4: Quantitative results of 3D segmentations bootstrapped from sparse annotations. a, Example images and ground-truth labels from each dataset and color-coding key. The total path length of the dense ground-truth skeletons in each training volume is indicated in parenthesis. **b,** Number of split and merge errors needing correction per skeleton path length to match the dense ground-truth skeletons. Lower scores are better. Each dataset had three separate tests for each amount of annotation (illustrated as three dots of the same color per row). Dense segmentations of a separate test volume (pseudo ground-truth) were first created by 2D->3D models trained on different amounts of initial ground truth annotations, which are sorted from bottom to top of the y-axis in order of increasing human annotation effort. Scores were then computed by comparing ground-truth annotations to segmentations produced by a 3D model trained on the pseudo ground-truth segmentation. 2D (or 3D) Dense refers to training the initial 2D->3D (or 3D) model on all available 2D (or 3D) annotations to generate pseudo ground-truth. GT Dense refers to directly training a 3D model on ground-truth annotations of the test volume to generate the segmentation of the initial training volume.